THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)*

Présentée et soutenue le *06/12/2018* **par :**

BASTIEN TAURAN

ON THE INTERACTION BETWEEN TRANSPORT PROTOCOLS AND LINK-LAYER RELIABILITY SCHEMES FOR SATELLITE MOBILE SERVICES

SUR L'INTERACTION ENTRE PROTOCOLES DE TRANSPORT ET FIABILISATION COUCHE LIAISON POUR SERVICES MOBILE SATELLITE

JURY

Eugen Dedu	Université de Franche-Comté	Rapporteur
Pascal Anelli	Université de la Réunion	Rapporteur
Thierry Gayraud	Université Paul Sabatier	Président / Examinateur
Caroline Bes	CNES	Examinatrice
Emmanuel Lochin	ISAE-SUPAERO	Directeur de thèse
Jérôme Lacan	ISAE-SUPAERO	Co-directeur de thèse

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture
Unité de Recherche :
TéSA - ISAE-SUPAERO
Directeur(s) de Thèse :
Emmanuel Lochin et *Jérôme Lacan*
Rapporteurs :
Eugen Dedu et *Pascal Anelli*

Remerciements

Cette thèse n'aurait jamais pu aboutir sans le soutien précieux de nombreuses personnes, qui m'ont encouragé et aidé durant ces trois années.

Je tiens tout d'abord à remercier mes directeurs de thèse: Emmanuel Lochin et Jérôme Lacan, qui m'ont guidé, posé les bonnes questions, toujours été à mon écoute et permis de constamment m'améliorer. Ils ont toujours été présents durant ces trois années, pour me donner des conseils, partager leur vision des problèmes rencontrés, ou réaliser des sessions d'*extreme writing*. Tout mon travail effectué durant ces trois ans n'aurait jamais pu être aussi efficace sans eux.

Je voudrais également remercier mes encadrants côté CNES et TAS: Nicolas Kuhn, Fabrice Arnal et Mathieu Gineste, ainsi que Emmanuel Dubois et Laurence Clarac, leurs connaissances très poussées du secteur spatial étant un atout non négligeable. Les réunions avec tout les encadrants ont toujours permis de soulever des questions pertinentes et de proposer des pistes de travail prometteuses.

Je remercie enfin les membres du jury de ma thèse pour avoir pris le temps de relire mon manuscrit de thèse, de proposer des corrections pertinentes et d'avoir posé des questions précises et intéressantes lors de ma soutenance.

Un grand merci également à tous mes collègues et amis à l'ISAE. Jonathan et ses bitcoins (ou ce qu'il en reste désormais...), Antoine et sa passion pour les pugs, Henrick et sa passion pour les langages purs (j'attends toujours que tu viennes nager avec moi !), Doriane et ses voyages, Fred et ses sessions de pole dance, Anaïs et ses applis Android, Clément et son watercooling, ainsi qu'Odile et Alain.

Je n'oublie pas non plus mes collègues du TéSA: Romain, Charles-Ugo, Sylvain, Selma, Raoul, Adrien, Barbara. Je ne peux pas citer tout le monde mais je ne vous oublie pas ! Je remercie aussi au TéSA Corinne et Isabelle, qui ont toujours été très efficaces sur le plan administratif et ont organisé de fantastiques séminaires avec tout le labo !

Mes amis hors du cadre du travail m'ont également permis de passer ces trois années dans les meilleures conditions. Je tiens à les remercier tous, ils se reconnaîtront !

Je tiens enfin à remercier le plus chaleureusement possible ma famille, en premier lieu mes parents qui ont été un soutient de taille et indéfectible pendant cette période, ainsi que les autres membres de ma famille !

Finalement, je tiens à remercier le CNES et Thales Alenia Space pour le financement de cette thèse.

Contents

List of Figures

List of Tables

List of Abbreviations

AQM	Active Queue Management
ARQ	Automatic Repeat reQuest
BBS	BBFrame by BBFrame Scheduling
BPS	BBS Periodic Scheduling
cwnd	congestion window
D-SACK	Duplicate SACK
DelAck	Delayed Acknowledgment
DT	Drop Tail
DUPACK	DUPlicate ACKnowledgment
ECC	Error Correcting Code
ECN	Explicit Congestion Notification
EXP-PF	Exponential PF
FEC	Forward Error Correction
FLN	Fat Long Network
GEO	Geostationnary Earth Orbit
HARQ	Hybrid Automatic Repeat reQuest
HSDPA	High Speed Downlink Packet Access
HSUPA	High Speed Uplink Packet Access
IP	Internet Protocol
ISL	Inter-Satellite Link
IW	Initial Window
LEO	Low Earth Orbit

LMS	Land Mobile Satellite
M-LWDF	Maximum Largest Weighted Delay First
MEO	Medium Earth Orbit
MOS	Mean Opinion Score
PEP	Performance Enhanced Proxy
PF	Proportional Fairness
PTO	Probe TimeOut
QoE	Quality of Experience
QoS	Quality of Service
RACK	Recent ACKnowledgment
RTO	Retransmission TimeOut
RTT	Round Trip Time
SACK	Selective Acknowledgment
TCP	Transmission Control Protocol
TLS	Tail Loss Probe
UBMT	Urgency Based Maximum Throughput
UDP	User Datagram Protocol
VOD	Video On Demand
VoIP	Voice over IP

Chapitre 1

Synthèse en français

1.1 Introduction

1.1.1 Contexte

L'universalité de l'accès Internet n'est pas encore une réalité. Il existe de nombreuses zones encore isolées où la fracture numérique ne sera pas résorbée avant des décennies. En effet, une solution terrestre peut dans ce cas être très coûteuse voire impossible à réaliser. Une solution possible à ce problème passe par l'utilisation de liens satellites, afin d'avoir une couverture globale au coût de nouvelles contraintes spécifiques à cet environnement.

Les constellations de satellite en orbite basse (LEO) ont des délais de transfert similaires à ceux des réseaux terrestres autorisant l'utilisation des protocoles de transport classiques, tels que TCP. Ce type d'environnement engendre cependant des délais variables causés par le mouvement des satellites, occasionnant des changements de chemins lors des transmissions. De plus, la traversée de l'atmosphère, l'environnement du terminal ainsi que de possible interférences rendent la transmission plus difficile, et les erreurs doivent être corrigées. Le lien le plus problématique dans notre cas est celui entre le dernier satellite sur la route du paquet et le récepteur au sol, que l'on suppose en mouvement. Ce lien est appelé lien *Land Mobile Satellite* (LMS) et a une qualité très variable dans le temps à cause des mouvements du récepteur au sol.

Pour compenser ce fort taux d'erreur sur le canal LMS, des mécanismes de fiabilisation doivent être introduits sur ce lien, dans le but de corriger les potentielles erreurs le plus rapidement possible, tout en optimisant la capacité du canal. Ces mécanismes ajoutent de la redondance sur l'information et des retransmissions dans les couches basses si nécessaire. L'impact de ces mécanismes sur les couches hautes, notamment TCP, a cependant besoin d'être correctement étudié. En effet, il est impossible de se passer des protocoles terrestres, conçus pour des délais faibles et des taux d'erreur faibles, car les nœuds terminaux ne sont pas conscients de la présence d'un segment spatial sur le chemin.

Nous étudions dans cette thèse l'impact des mécanismes de fiabilisation des couches basses sur les performances des protocoles de transport, notamment TCP, dans le cadre de flots longs. Nous expliquons dans un premier temps les raisons de cette mauvaise performance, puis donnons des solutions pour la contrer. Enfin nous apportons des propositions d'amélioration afin d'augmenter les performances des mécanismes de fiabilisation et ainsi optimiser la capacité du canal LMS tout en n'ayant pas d'effets négatifs sur les couches hautes.

1.1.2 Contributions et organisation

Cette thèse étudie l'impact des mécanismes de fiabilisation introduits sur le canal LMS sur les performances des protocoles de transport, notamment TCP. Elle propose ensuite des solutions pour minimiser cet impact puis optimiser ces mécanismes.

Dans un premier temps, nous analysons l'impact d'un mécanisme de fiabilisation performant dans ce contexte, nommé HARQ, sur les performances de TCP. Nous en concluons que la mauvaise performance de TCP est due aux paquets délivrés dans le désordre par HARQ aux couches hautes, provoquant des retransmissions abusives. Pour corriger cela, nous proposons d'introduire un mécanisme de réordonnancement en sortie d'HARQ, afin de délivrer les paquets dans l'ordre à TCP. Ce mécanisme sera détaillé en Section 1.2.

Dans un deuxième temps, nous nous attarderons sur l'impact de l'option *Delayed Acknowledgment* (DelAck) sur les performances de TCP, dans notre contexte de constellation LEO de satellites. Cette option améliorant les performances sur les réseaux terrestres, elle est souvent activée par défaut dans les systèmes. Cependant nous montrons en Section 1.3 que celle-ci peut être contreproductive dans notre scénario et son activation devrait dépendre des variantes de TCP utilisées.

Enfin, en Section 1.4, nous proposons d'ajouter un mécanisme d'ordonnancement entre les utilisateurs afin d'optimiser la capacité du canal LMS. Cette étude se fait avec deux types de trafic : VoIP et TCP, le premier permettant d'introduire une nouvelle politique d'ordonnancement appelée CoDeS, basée sur un ordonnanceur de type *Proportional Fairness*, le deuxième démontrant la bonne performance d'un simple ordonnanceur *Proportional Fairness*. Dans ces deux cas, ces propositions permettent d'optimiser la capacité du canal en améliorant la performance des mécanismes de fiabilisation.

Toutes ces contributions permettent d'augmenter la performance des protocoles de transport et ainsi que la Qualité d'Expérience (QoE) et la satisfaction des utilisateurs. La performance globale du réseau est également grandement améliorée.

1.2 Impact du désordonnancement sur les performances de TCP

Dans cette première partie, nous nous intéressons à l'impact des mécanismes de fiabilisation des couches basses sur les performances de TCP. Nous définissons dans un premier temps précisément la constellation de satellites, les mécanismes de fiabilisation utilisés, et les variantes de TCP utilisées, pour ensuite analyser les causes de la mauvaise performance de TCP, et enfin proposer une solution efficace.

1.2.1 Scénario

Nous prenons l'exemple d'une constellation composée de 66 satellites en orbite basse, avec routage à bord et à une altitude de 800 km. Ce genre de constellation, similaire aux constellations réellement déployées ou en cours de déploiement, permet d'assurer une couverture totale de n'importe quelle région de la Terre, à n'importe quel moment.

Le lien entre le dernier satellite sur le trajet du message et le récepteur mobile au sol est un lien affecté par un modèle de canal de transmission appelé *Land Mobile Satellite* (LMS) [1, 2, 3], comme montré en Figure 1.1, fonctionnant sur des bandes de fréquences basses. Un satellite peut servir plusieurs utilisateurs, toutefois, dans cette Section, nous supposerons que nous avons qu'un seul flot transmis dans la constellation et donc un seul utilisateur au sol. L'extension à plusieurs utilisateurs sera faite en Section 1.4.

C'est sur le lien LMS que la qualité de la liaison est la moins bonne, à cause de la traversée de l'atmosphère, de l'environnement du récepteur et de possibles interférences. Son importante et sa rapide variation en termes de capacité en fait le goulot d'étranglement du réseau. Aussi, il est nécessaire d'inclure des mécanismes de fiabilisation efficaces sur ce lien.

Un des candidats les plus efficaces dans ce contexte est *Hybrid Automatic Repeat reQuest* (HARQ), permettant de corriger les erreurs le plus rapidement possible tout en optimisant l'utilisation de la capacité du canal. Ce mécanisme est compatible avec notre scénario car il adapte le nombre de symboles envoyés à la capacité du canal, qui varie beaucoup dans notre cas. Plus particulièrement, nous présentons une amélioration de HARQ de type II (présenté en Figure 1.2), appelée Adaptive-HARQ [4]. Cette version de HARQ envoie dans un premier temps les bits utiles avec quelques bits de redondance. Le récepteur décide ensuite en utilisant l'Information Mutuelle [5] si le paquet peut être décodé ou non. Dans ce dernier cas, il va demander

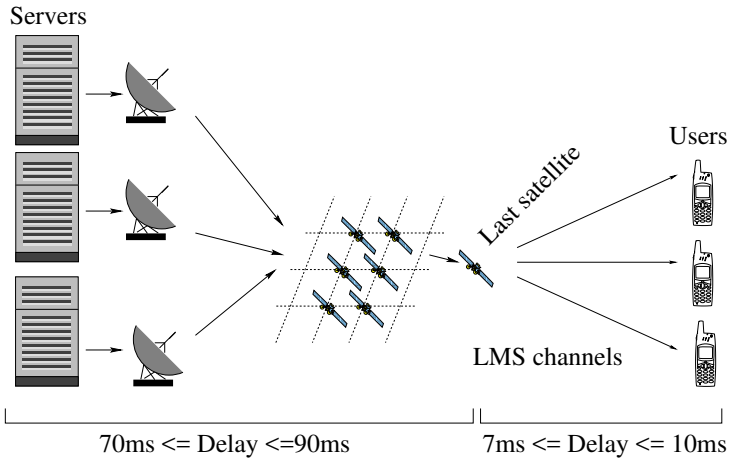


FIGURE 1.1 – Environnement satellite avec le canal LMS

au dernier satellite d'envoyer des bits de redondance supplémentaires, et ce ainsi jusqu'à ce que le paquet soit décodé ou que l'on atteigne 3 retransmissions, auquel cas le paquet est considéré perdu. Ce mécanisme permet ainsi d'adapter le nombre de retransmissions à la capacité du canal LMS.

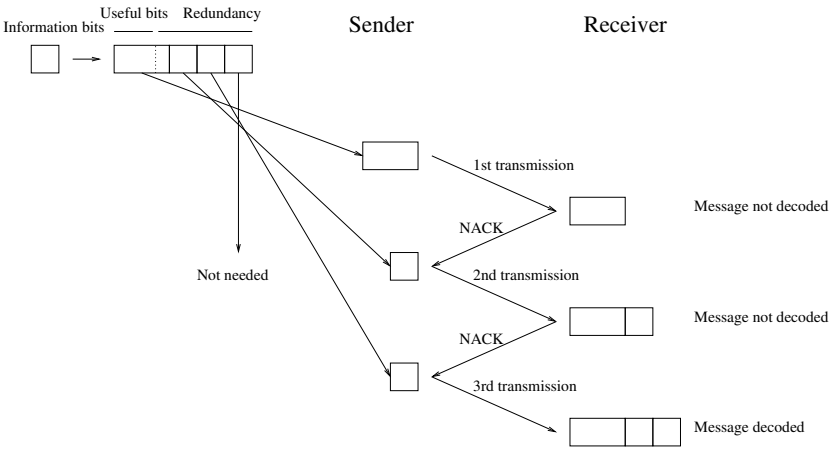


FIGURE 1.2 – HARQ type II

Nous utilisons le simulateur network simulator 2 (*ns*-2) [6] pour nos expérimentations. Ce simulateur, largement utilisé par la communauté scientifique, est *open-source* et permet de simuler des constellations de satellites. Nous y avons implémenté le mécanisme HARQ qui n'est pas présent par défaut dans le simulateur. Concernant le canal LMS, nous prenons le modèle présenté par [3], et utilisons des traces temporelles fournies par le CNES représentant l'évolution du canal. Dans *ns*-2, nous avons fixé la bande passante du lien à 50 Mb/s, qui est une valeur acceptable pour le type de canal choisi. Toutefois les résultats obtenus avec cette valeur

de canal ne sont pas un cas particulier, et des simulations avec d'autres ordres de grandeur de bande passante ont montré des résultats similaires. Nous précisons enfin que cette valeurs de 50 Mb/s doit être partagée par tous les utilisateurs au sol.

Concernant les protocoles de transport, nous utilisons TCP, en qualité de protocole dominant d'Internet. Plus précisément, nous utilisons deux de ses principales variantes :

- TCP NewReno [7], qui est l'une des variantes les plus simples de TCP. Cette version utilise un algorithme de contrôle de congestion peu complexe. TCP NewReno fait office de variante de référence pour nos simulations ;
- CUBIC [8], qui est quant à lui une variante créée pour fonctionner sur les réseaux à fort produit bande passante - délai. Sa fenêtre de congestion est plus agressive et ses bonnes performances l'ont amené à devenir la variante de TCP par défaut dans de nombreux systèmes (GNU-Linux OS, MacOS, Windows OS), motivant sa sélection dans notre étude.

Dans nos analyses, nous allons nous intéresser à deux métriques de TCP permettant de détecter des pertes : DUPACK (au sein du flot) et RTO (en fin de flot). Plusieurs options de TCP sont également testées afin d'en comprendre leur impact, telles que D-SACK, ou DelAck. Tous ces paramètres sont présentés dans la Section 3.4.

Pour finir, chaque simulation est composée d'un seul flot TCP, envoyant des données pendant 600 s, permettant d'atteindre le régime établi de TCP et de pouvoir constater l'impact de l'environnement satellite et des mécanismes de fiabilisation. Enfin, la qualité moyenne du canal varie d'une simulation à l'autre autour d'un SNR de référence allant de 7 dB à 13 dB.

1.2.2 Impact du désordonnancement

La Table 4.1 (page 63) donne les performances de TCP NewReno et CUBIC. Elle relève la mauvaise performance de TCP en termes de débit utile. En effet, nous avons un débit utile de l'ordre de quelques centaines de kb par seconde, alors que la topologie étudiée permettrait d'atteindre des débits jusqu'à 40 Mb/s pour un lien sans pertes (résultat obtenu par simulation mais non présenté ici).

Pour expliquer cette mauvaise performance, il faut regarder les autres métriques, toujours affichées dans la Table 4.1 (page 63). Nous constatons alors une proportion élevée de DUPACK et de retransmissions abusives (spurious retransmissions, représentant les paquets retransmis par TCP alors qu'ils ont bien été reçus lors du

1er envoi). Cette grande proportion de DUPACK est directement liée aux délais variables causés par la constellation et HARQ : les paquets arrivent dans le désordre au niveau de la couche transport, et TCP considère les paquets retardés comme perdus à cause de congestion dans le réseau, engendrant une retransmission abusive et une baisse de la fenêtre de congestion inutile.

L'envoi des paquets de façon désordonnée aux couches supérieures est donc la cause de la mauvaise performance de TCP. Il est donc nécessaire de trouver une solution pour compenser cet effet.

1.2.3 Solution proposée

Une solution possible est d'ajouter un mécanisme de réordonnancement en sortie de HARQ, comme présenté en Figure 1.3. Le fonctionnement détaillé de ce mécanisme est donnée en Section 4.4 (page 64).

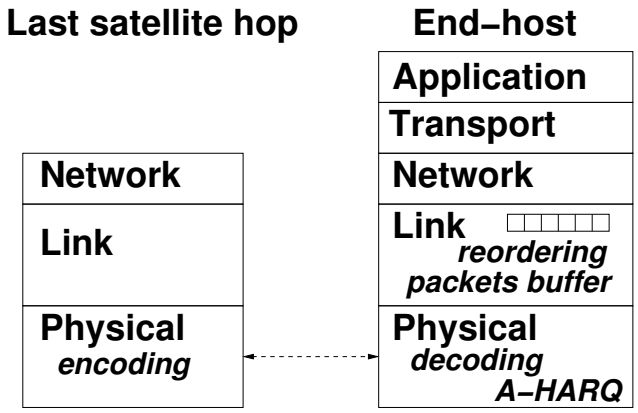


FIGURE 1.3 – Proposition d'ajout de mécanisme de réordonnancement après HARQ

Les performances obtenues par TCP avec ce mécanisme sont données dans la Table 4.2 (page 67). On observe un gain important de débit utile pour les deux variantes de TCP et ce, quel que soit le SNR de référence (au sens défini dans [9]), comme montré en Figure 1.4. De plus, comme espéré, la proportion de DUPACK diminue avec l'ajout de ce mécanisme.

Cependant, la Figure 1.4 montre un impact négatif et non négligeable de DelAck sur la performance de TCP avec le mécanisme de réordonnancement. Cet impact est étudié en détail en Section 1.3.

L'analyse approfondie des résultats montre une plus grande utilisation du canal LMS, et une augmentation de la fenêtre de congestion moyenne, ce qui est corrélé à l'augmentation du débit utile. Finalement, on remarque que, notamment pour des bons SNR, les pertes de paquets dans le réseau sont principalement dues à de la

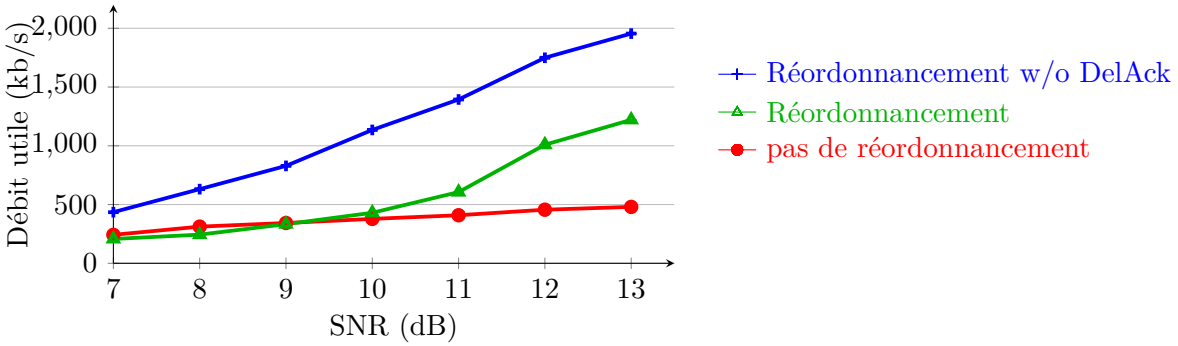


FIGURE 1.4 – Impact du mécanisme de réordonnancement sur le débit utile (CUBIC)

congestion, permettant à TCP de fonctionner dans ses conditions nominales et de montrer de bonnes performances.

En conclusion, l'étude que nous avons conduite avec Adaptive-HARQ peut être étendue à toutes les variantes de HARQ puisqu'elles transmettent toutes des paquets dans le désordre aux couches supérieures. L'ajout d'un mécanisme de réordonnancement conjointement à HARQ, et au niveau de la couche liaison en général, devrait donc être systématique afin de toujours maximiser le débit utile et la satisfaction des utilisateurs.

1.3 Étude de DelAck

Nous avons relevé dans la Section 1.2 que TCP *Delayed Acknowledgment* (DelAck) avait un impact non négligeable sur les performances de TCP lors de transmissions via une constellation LEO de satellites. Nous allons dans cette section tout d'abord présenter cette option de TCP, puis étudier en détail son impact dans notre contexte.

1.3.1 Présentation de DelAck

TCP *Delayed Acknowledgment* (DelAck) [10] est une option permettant d'améliorer les performances de TCP. DelAck réduit le nombre d'acquittements et ainsi la charge sur la voie retour et permet également de réduire la charge de calcul des processeurs [11]. Sur les réseaux terrestres, ces propriétés permettent d'améliorer la performance de TCP, ce qui a amené aujourd'hui à son activation par défaut dans la plupart des systèmes.

DelAck permet de combiner deux paquets reçus pour ne former qu'un seul acquittement, lorsque les deux conditions suivantes sont réunies :

- les paquets doivent arriver dans l'ordre. Si cette condition n'est pas respectée, TCP envoie directement un acquittement pour le paquet désordonné, pour permettre à l'émetteur de s'adapter le plus vite possible à une potentielle perte ;
- les deux paquets doivent arriver dans un intervalle de temps fixé, souvent de 40 ms, mais peut parfois être plus grand comme 200 ms avec Windows OS. C'est un compromis permettant d'attendre assez longtemps le second paquet sans risquer de timeout par TCP (RTO).

Il est remarquable que dans notre contexte, avec des paquets désordonnés et des longs délais, les conditions ne sont pas optimales pour le bon fonctionnement de DelAck. Dans la suite, nous allons étudier la performance de cette option, tout d'abord sans le mécanisme de réordonnancement présenté en Section 1.2, puis avec ce mécanisme.

1.3.2 Impact sans réordonnancement

Nous réutilisons les mêmes conditions expérimentales qu'en Section 1.2, c'est-à-dire une constellation LEO de satellites, avec un canal LMS sur le dernier lien et un seul flot TCP transmis à travers ce réseau. Nous avons relevé toutefois dans des simulations additionnelles que nous obtenions des résultats similaires avec plusieurs flots en parallèle, permettant d'étendre les résultats présentés ici au cas de plusieurs utilisateurs en parallèle. Enfin, nous gardons la valeur par défaut de *ns*-2 pour l'intervalle de temps maximal entre deux paquets formant un DelAck : 40 ms.

Nous observons les performances de TCP en fonction de l'activation de DelAck en Figure 1.5, et remarquons deux évolutions différentes en fonction de la variante de TCP utilisée : nous avons une augmentation du débit utile avec TCP NewReno mais une performance quasi inchangée avec CUBIC.

Des mesures ont également été menées avec TCP Hybla [12], protocole développé pour les liens GEO, et qui peut être une variante intéressante à analyser ici. Dans ce cas, l'activation de DelAck réduit fortement les performances de TCP, avec une chute du débit utile allant jusqu'à 40 %.

L'explication de ces résultats hétérogènes est à trouver dans la façon dont la fenêtre de congestion évolue avec DelAck. En effet, celle-ci est mise à jour lorsqu'un

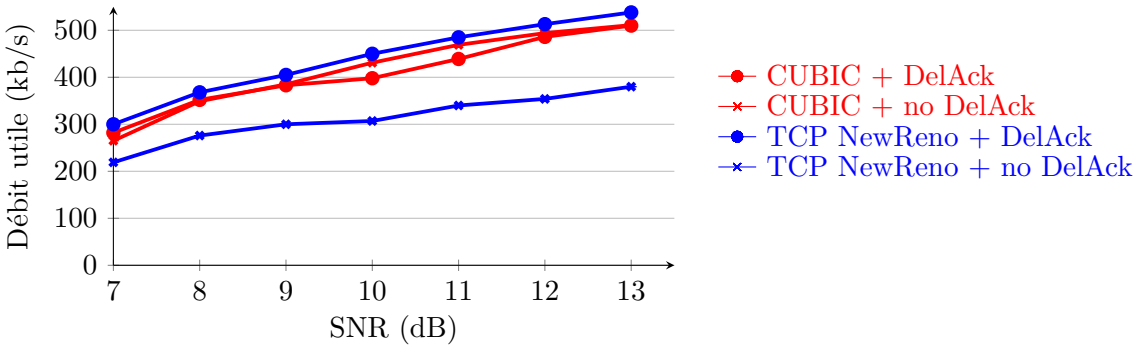


FIGURE 1.5 – Impact de DelAck sur le débit utile

acquittement est reçu, et l'activation de DelAck, retardant la réception des acquittements, ralentit la croissance de la fenêtre, notamment en phase de *Slow Start*. Une illustration montrant ce ralentissement est donnée en Figure 5.7 (page 78).

Ainsi les variantes de TCP retransmettant beaucoup vont être négativement impactées par cette croissance plus faible de la fenêtre, contrebalançant les avantages apportés par DelAck. On remarque dans la Table 5.1 (page 79) que CUBIC retransmet plus que TCP NewReno, et que TCP Hybla retransmet plus que les deux autres variantes. C'est donc cette dernière variante qui va être la plus impactée négativement par DelAck, amenant à une baisse de performance. Au contraire, le faible nombre de retransmissions avec TCP NewReno lui permet de toujours profiter des avantages de DelAck et ainsi d'améliorer ses performances.

En conclusion, lorsque nous n'avons pas de mécanisme de réordonnancement après HARQ, l'activation de DelAck est conseillée seulement pour les variantes de TCP peu agressives, tel que TCP NewReno. Pour les variantes plus agressives, telles que CUBIC ou TCP Hybla, nous recommandons de désactiver si possible DelAck pour optimiser les performances de TCP.

1.3.3 Impact avec réordonnancement

Comme nous l'avons vu en Section 1.2, l'ajout du mécanisme de réordonnancement en sortie de HARQ permet d'améliorer les performances de TCP au prix d'un délai de transfert plus élevé. Ce changement nous amène à nous demander si l'utilisation d'un mécanisme ajoutant du délai aux acquittements comme DelAck est compatible ou non avec notre buffer de réordonnancement. Les résultats sont donnés pour TCP NewReno en Figure 1.6 et CUBIC en Figure 1.7.

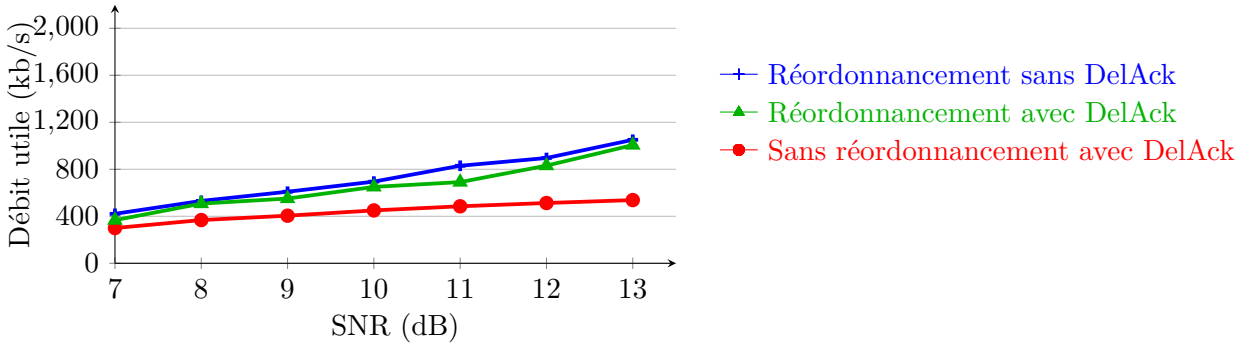


FIGURE 1.6 – Impact de DelAck sur la performance de TCP avec le mécanisme de réordonnancement (TCP NewReno)

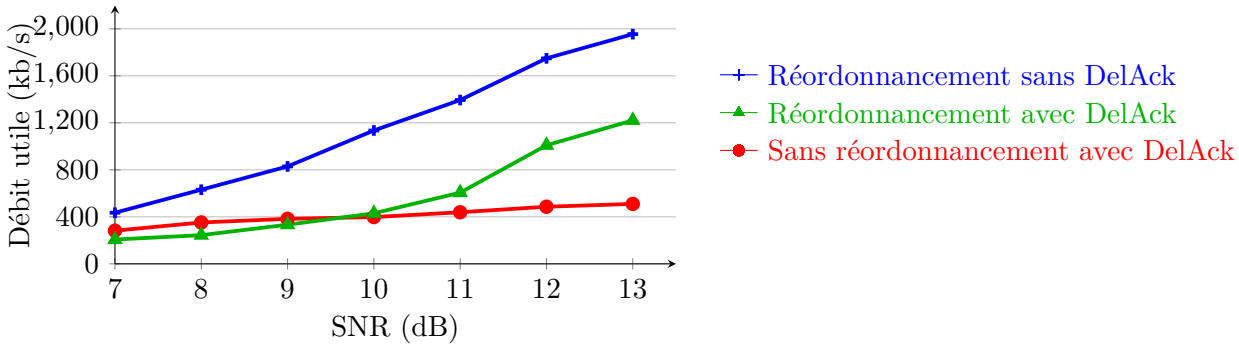


FIGURE 1.7 – Impact de DelAck sur la performance de TCP avec le mécanisme de réordonnancement (CUBIC)

Nous pouvons désormais observer que, quelle que soit la variante de TCP utilisée, l'utilisation conjointe de DelAck et de notre mécanisme de réordonnancement diminue le débit utile de l'utilisateur, notamment avec CUBIC. Cette tendance est également observée avec TCP Hybla. On peut extraire deux conclusions de ces courbes :

- lorsque l'on suppose que le mécanisme de réordonnancement est toujours activé, ajouter DelAck diminue la performance de TCP ;
- cependant, si l'on suppose que DelAck est toujours activé, ajouter le mécanisme de réordonnancement améliore les performances de TCP, comme on peut le voir sur la Figure 1.7.

La solution optimale est ainsi de réordonnancer les paquets et de désactiver DelAck, quand cela est possible.

La baisse de performance de TCP lors de l'activation de DelAck dans ce cas diffère des résultats obtenus sans le mécanisme de réordonnancement. La diminution

de débit utile avec DelAck est désormais due à l'augmentation du délai de transmission de bout en bout. En effet, en plus du délai relatif à la topologie, HARQ, et le mécanisme de réordonnancement augmentent le délai de transfert. L'ajout de DelAck augmente encore plus ce délai déjà élevé, accroissant la probabilité de timeout par TCP.

Nous avons ainsi mesuré, comme montré en Figure 5.11 (page 83), que l'activation de DelAck rajoute un délai d'au moins 10 ms pour 10 % des paquets reçus, impliquant une augmentation de la proportion de RTO par TCP. Cette augmentation est renforcée par la diminution de la valeur du *timer* de RTO. Cette valeur est une fonction du RTT et de sa variation. Or DelAck lisse les mesures de RTT, diminuant la variation de RTT, et ainsi la valeur du *timer* de RTO. Finalement les paquets mettent plus de temps à arriver alors que le *timer* expire plus vite. L'augmentation du nombre de RTO est montrée en Figure 1.8.

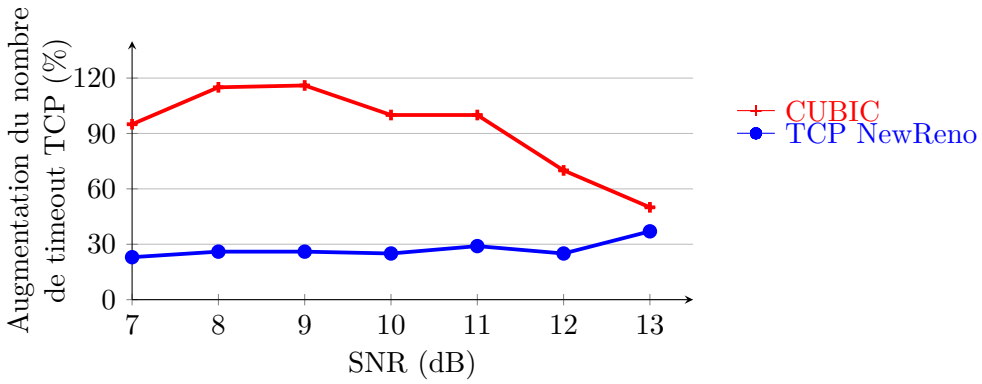


FIGURE 1.8 – Impact de l'activation de DelAck sur le nombre de retransmissions quand le mécanisme de réordonnancement est activé

L'option DelAck de TCP a donc des impacts très différents en fonction de la variante de TCP utilisée et de l'activation ou non du mécanisme de réordonnancement. Sans mécanisme de réordonnancement, le gain de performance avec DelAck ne sera possible que pour les versions les moins agressives de TCP, et, seulement pour ces variantes, il est recommandé d'activer cette option. Avec le mécanisme de réordonnancement, l'activation de DelAck est toujours néfaste pour la performance de TCP, et cette option devrait toujours être désactivée.

1.4 Ordonnancement des flots afin d'optimiser la capacité du canal LMS

Dans cette dernière partie, nous analysons l'impact de l'ajout de mécanisme d'ordonnancement entre les utilisateurs avant l'envoi des paquets sur le canal LMS. En effet, le canal LMS est le goulot d'étranglement du réseau, et l'optimisation de sa capacité permet d'augmenter les performances globales de la transmission. Le but d'un tel ordonnanceur est d'envoyer les paquets vers un utilisateur uniquement quand il a une bonne qualité de canal LMS. Ainsi, le temps de reconstruction des paquets par HARQ est plus court, et la capacité de canal utilisée par paquet est plus faible.

1.4.1 Contexte et scénario

Le mécanisme HARQ a besoin de plus ou moins de temps et de capacité de canal pour reconstruire un paquet, en fonction de la qualité du canal au moment de l'envoi. Dans le cas de plusieurs utilisateurs ayant le même dernier satellite sur leur chemin, et un récepteur différent au sol, si aucune politique d'ordonnancement n'est mise en place, chaque utilisateur aura un canal qui peut être soit de bonne qualité soit très mauvais, et un temps de reconstruction par HARQ très variable.

L'ajout d'un mécanisme d'ordonnancement entre les paquets va pouvoir permettre de favoriser l'envoi de paquets uniquement vers les utilisateurs ayant un bon canal, les autres devant attendre que leurs conditions s'améliorent. Cela permet ainsi de diminuer la capacité du canal utilisée par paquets ainsi que le temps de reconstruction des paquets par HARQ. Il nous faut donc trouver un mécanisme permettant de favoriser les utilisateurs ayant des bons canaux, tout en évitant des situations de famine pour les autres, et sans impact négatif sur les protocoles de transport.

Dans cette étude, nous supposons que la qualité du canal est connue instantanément par le satellite. En réalité, l'estimation prendrait aux alentours de 10 ms pour arriver au satellite, mais durant cette courte période, la capacité du canal n'évolue pas suffisamment pour avoir des valeurs totalement différentes par rapport au moment de la mesure. L'estimation instantanée du canal peut donc être considérée comme une bonne hypothèse.

Une première série de mesures avec Matlab a montré que si on ne considère que les couches basses, l'envoi des paquets vers l'utilisateur ayant le meilleur canal permet d'envoyer plus de données utiles, et ainsi d'optimiser la capacité du canal,

comme montré en Figure 1.9. Dans cette figure, nous supposons que tous les utilisateurs ont constamment des paquets à envoyer, et que le choix de l'utilisateur se fait soit de façon aléatoire, soit en sélectionnant celui avec le meilleur canal.

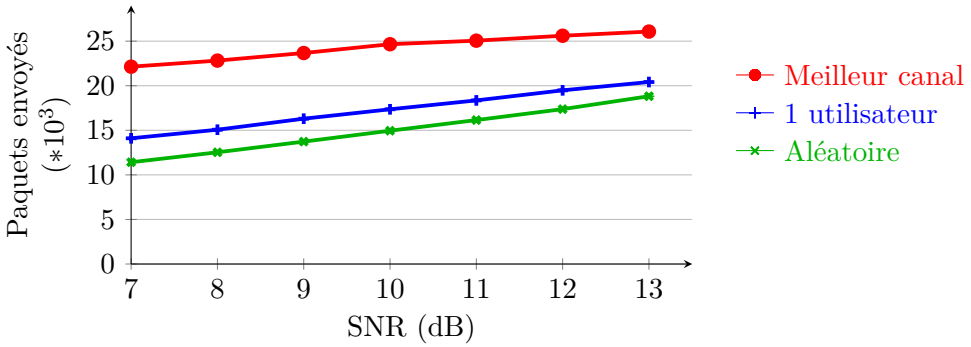


FIGURE 1.9 – Nombre de paquets envoyés par les couches basses dans différents scénarios

Concernant les ordonnanceurs possibles, nous avons relevé la liste des principaux candidats adaptés à notre scénario. Ils sont développés pour des types de trafic différents et ont des complexités variables. Ils sont présentés dans la Table 1.1.

Algorithme	Métriques	Trafic	Complexité
PF [13]	Débit	best-effort	Très faible
M-LWDF [14]	Débit et temps dans la file	Temps réel	Faible
EXP-PF [15]	Débit et temps dans la file	Temps réel et BE	Faible
UBMT [16]	Débit et temps dans la file	Tous	Moyen
BBS et BPS [17]	Débit ou temps dans la file	Dépend de la métrique	Élevé

Tableau 1.1 – Présentation des principaux ordonnanceurs

Un des algorithmes les plus simples en termes de complexité est *Proportional Fairness* (PF). Il utilise le débit du lien LMS pour effectuer son ordonnancement. Cet ordonnanceur utilise autant de files d'attentes qu'il y a de destinations, et va calculer la valeur $f_i(r)$ suivante pour chacune d'entre elles, chaque fois que le canal LMS est disponible pour envoyer un nouveau paquet :

$$f_i(r) = \frac{r_i(t)}{\overline{r_i}(t)} \tag{1.1}$$

Dans cette équation, $r_i(t)$ représente le débit du canal au moment t et $\overline{r_i}(t)$ la moyenne de ce débit. La file d'attente qui est choisie est celle ayant la plus grande valeur de $f_i(r)$, et le paquet en tête de cette file est transmis sur le canal LMS. L'avantage d'un tel algorithme est de permettre de sélectionner l'utilisateur avec le

meilleur canal tout en évitant qu'il n'envoie pendant trop longtemps, permettant ainsi d'éviter des situations de famine pour les autres utilisateurs. De plus, dans notre contexte, le débit sur le canal est directement lié à la capacité de celui-ci, et l'algorithme PF peut être adapté pour fonctionner en utilisant les capacités du canal LMS directement.

La taille des files d'attentes pour PF a été dimensionnée afin de minimiser l'espace de stockage, de facto limité dans un environnement satellite. Les auteurs de [18] ont établi la règle suivante pour le calcul de la capacité de stockage nécessaire dans une file d'attente :

$$B = \frac{\overline{RTT} * C}{\sqrt{n}} \tag{1.2}$$

où $\overline{RTT}$ correspond au temps d'aller-retour moyen, C à la bande passante du lien et n le nombre d'utilisateurs en parallèle. La capacité totale de stockage étant de B et l'ordonnanceur ayant n files d'attente, chaque file aura une capacité de B/n.

Nous montrons dans cette section que bien que PF soit adapté à du trafic best-effort, le simple ajout d'une politique de gestion des files d'attente le rend compatible avec un trafic VoIP. Nous appelons ce nouveau mécanisme *Controlled Delay Scheduler* (CoDeS) et en détaillons le fonctionnement dans la section suivante.

Finalement, à des fins comparatives, les performances de PF et CoDeS vont être évaluées conjointement à d'autres politiques n'ayant pas pour but d'optimiser la capacité du canal LMS :

- Round Robin (RR) : cet ordonnanceur sélectionne les files à tour de rôle, un paquet étant transmis par file à chaque fois ;
- DropTail Petit buffer (DT_S) : une seule file d'attente avec une politique FIFO. La taille de cette file est la même qu'une file de PF ou RR, c'est-à-dire B/n ;
- DropTail Grand buffer (DT_H) : une seule file avec une politique FIFO, mais avec une file de taille B cette fois.

1.4.2 Cas de trafic VoIP

Avant de lancer nos tests avec TCP, nous nous intéressons dans un premier temps au cas de trafic VoIP, pour lequel le but est de maximiser la Qualité d'Expérience (QoE) des utilisateurs en minimisant le délai de transmission, la gigue et le taux

d'erreur. L'utilisation d'UDP nous permet une analyse plus simple, paquet par paquet, des performances des ordonnanceurs. Nous montrons qu'un ordonnanceur PF associé à un système de gestion des files d'attente permet d'atteindre des bonnes QoE.

Les caractéristiques du trafic VoIP sont donnés dans la Table 1.2. HARQ utilisant des paquets d'une taille de 1115 octets, les paquets VoIP sont complétés avec des zéros pour obtenir la taille souhaitée, entraînant un capacité théorique du canal LMS de 9.4 Mb/s au lieu des 50 Mb/s sans remplissage.

Propriété	Valeur
Débit	64 kb/s
Distribution (temps d'émission et repos)	Pareto
Durée d'émission	500 ms
Durée de repos	50 ms
Taille des paquets	210 octets

Tableau 1.2 – Caractéristiques du trafic VoIP choisi

La norme ITU-T G114 [19] donne également des limites sur les principales métriques à respecter pour que les utilisateurs aient une bonne QoE :

- le délai doit être inférieur à 400 ms, mais cela correspond à une valeur maximale. Une valeur acceptable est plutôt de l'ordre de 200 ms ;
- la gigue doit être la plus faible possible ;
- le taux de pertes doit être inférieur à 3 %.

Ces restrictions sont la condition d'une bonne QoE. Il faut ajouter à cela que les utilisateurs, conscients de la présence d'un lien spatial sur le chemin, peuvent tolérer des conditions légèrement dégradées. Cette tolérance correspond à l'*Advantage Factor* utilisé par la norme ITU-T G114 [19] pour calculer la QoE des utilisateurs satellite. En effet, la QoE peut être quantifiée en utilisant le *Mean Opinion Score* (MOS), qui suit la norme ITU-T G114 [19]. Cette mesure donne une valeur entre 1 et 5 retranscrivant la qualité de la communication, en prenant en compte le délai, la gigue, le taux d'erreur. Son fonctionnement est détaillé dans la Section 3.2.

Nous menons plusieurs simulations avec *ns*-2, dans lequel les ordonnanceurs choisis ont été implémentés, et nous faisons varier le nombre d'utilisateurs de 25 à 200, pour simuler différentes charges sur le lien LMS. L'impact des ordonnanceurs sur le débit de la communication est donné en Figure 1.10. On observe un

meilleur débit avec PF, lié à la meilleure utilisation du canal LMS. Cette meilleure performance est confirmée par des meilleurs résultats concernant le nombre de retransmissions par HARQ (Table 6.3, page 98). Cela permet d'augmenter la capacité spectrale du canal : pour un même nombre de bits envoyés, on peut transmettre plus de bits utiles avec PF (Figure 6.9, page 99).

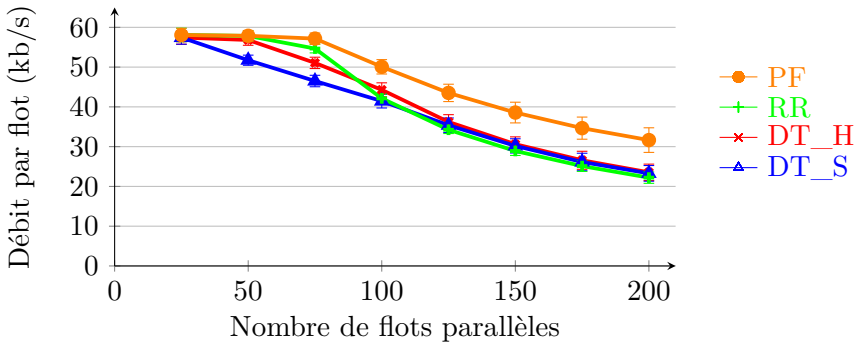


FIGURE 1.10 – Débit obtenu avec les différentes politiques d'ordonnancement

Les courbes montrant les résultats pour les différentes métriques utiles dans notre scénario sont présentées dans la Section 6.3. On observe de bons résultats pour PF : un plus faible taux d'erreur que les autres politiques, et un délai de transmission acceptable. Toutefois, PF souffre d'une forte gigue causée par la forte variation de la taille des files d'attente qui dont liées à l'évolution du canal LMS. Bien évidemment, cette gigue pénalise grandement la satisfaction des utilisateurs.

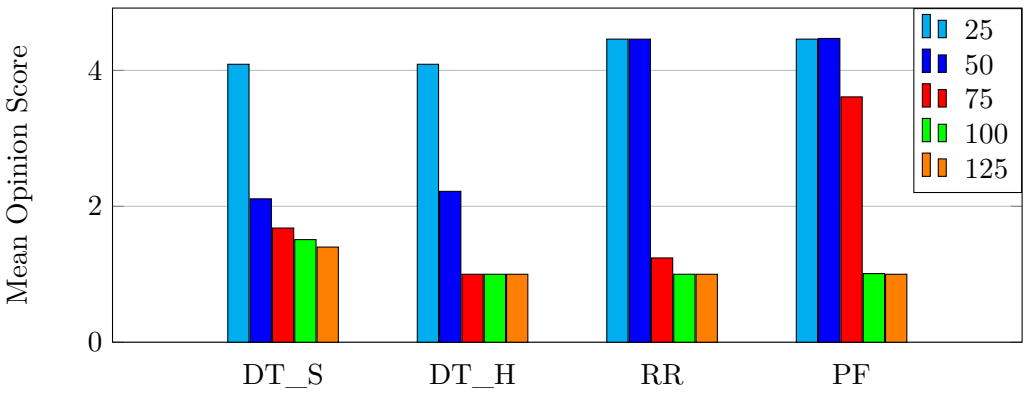


FIGURE 1.11 – *Mean Opinion Score* des politiques d'ordonnancement en fonction du nombre de flots

La Figure 1.11 présente le MOS calculé pour les politiques testées. Bien que PF montre les meilleures performances, à partir de 75 utilisateurs, la qualité de la transmission devient très mauvaise, notamment à cause de l'importance de la gigue.

D'un autre côté, nous observons qu'un certain nombre de paquets arrivés à destination sont jetés par la couche application car arrivés trop tard et donc considérés comme obsolètes par le codec VoIP, alors que d'autres paquets sont jetés par les files d'attente des ordonnanceurs car celles-ci sont pleines. Cependant, ces paquets auraient potentiellement pu être utilisables par la couche transport.

Ces constats nous ont mené à réfléchir à une politique de gestion de file d'attente conjointement à PF. Les paquets dans les files d'attente ont désormais un temps d'attente maximal au-delà duquel ils seront jetés, libérant de la place pour de nouveaux paquets. Cela permet à la couche application de recevoir des paquets en moyenne plus récents, ainsi que de réduire le délai de transfert et la gigue. Nous appelons ce nouveau mécanisme *Controlled Delay Scheduler* (CoDeS).

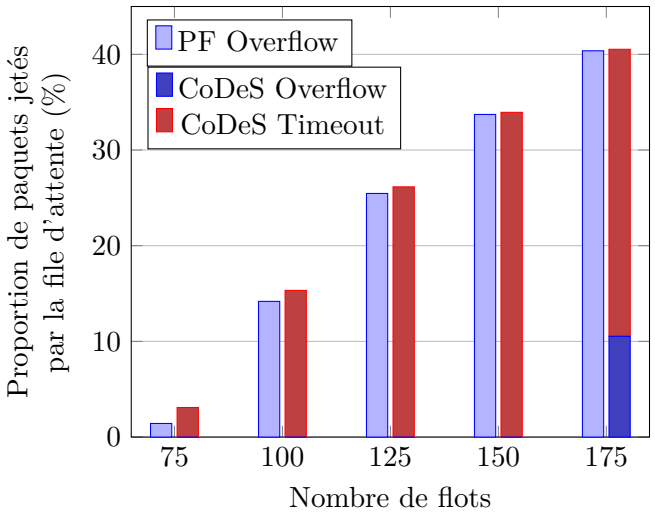


FIGURE 1.12 – Comparaison de la cause des rejets dans les files d'attente avec PF et CoDeS

Dans un premier temps, comme présenté dans la Figure 1.12, nous observons comme prévu que CoDeS ne change pas significativement le nombre de paquets jetés par la file d'attente, seule la cause de ces rejets est différente (*i.e. timeout* et non *overflow*).

Concernant les autres métriques, nous observons que CoDeS diminue le délai de transmission et la gigue, sans impacter les autres métriques. Nous avons également mesuré que la valeur optimale du *timer* dans les files d'attentes était entre 50 ms et 100 ms, ce qui permet de diminuer au maximum la gigue sans impacter sur le taux d'erreur. Nous prendrons dans la suite des travaux une valeur de 100 ms. Finalement le MOS est amélioré par rapport à PF, comme montré sur la Figure 1.13.

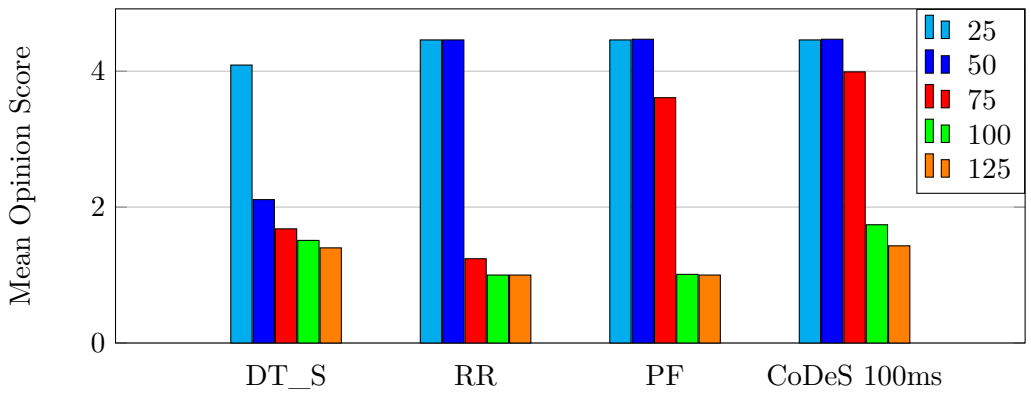


FIGURE 1.13 – *Mean Opinion Score* de DT_S, RR, PF et CoDeS 100ms

Le cas du trafic VoIP a permis de mettre en avant une nouvelle politique d'ordonnancement, appelée CoDeS, alliant un ordonnanceur PF et une gestion de la file d'attente. Cette politique permet d'optimiser la capacité du canal LMS, tout en limitant le délai de transmission et la gigue, permettant aux utilisateurs d'obtenir une bonne QoE.

1.4.3 Cas de trafic TCP

Nous nous intéressons maintenant au cas de trafic TCP. Nous cherchons à vérifier si les conclusions obtenues avec du trafic VoIP sont toujours valables. Les contraintes sont toutefois différentes ici : les paquets doivent tous arriver à la couche application sans erreurs, et le débit utile de la transmission par utilisateur doit être le plus important possible. Nous étudierons deux cas de figure, en fonction de l'ajout ou non du mécanisme de réordonnancement présenté dans la Section 1.2, en comparant les résultats obtenus avec les différents ordonnanceurs aux performances de TCP avec DT_H, correspondant à une file d'attente FIFO de taille classique.

Dans le premier cas, sans mécanisme de réordonnancement, nous obtenons les gains de débit utile par rapport à DT_H qui sont présentés en Figure 1.14. On observe des bonnes performances de PF, et CoDeS dans une moindre mesure, mais également de RR pour un faible nombre d'utilisateurs, qui parfois a les meilleures performances parmi toutes les politiques bien qu'elle n'optimise pas la capacité du canal LMS.

La moins bonne performance de CoDeS par rapport à PF s'explique par le fonctionnement de TCP, qui a déjà des mécanismes adaptant son taux d'émission de paquets à la congestion. Rajouter un nouveau mécanisme de gestion de files tels

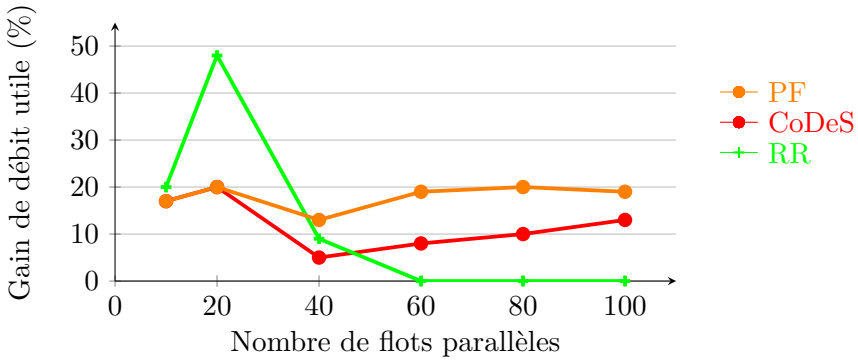


FIGURE 1.14 – Gain de débit utile par utilisateur pour différentes politiques, par rapport à DT_H (CUBIC, pas de réordonnancement)

que CoDeS est ici redondant avec le contrôle de congestion de TCP, et finalement contreproductif. De plus, le trafic best-effort peut supporter des délais de transmission plus longs, contrairement au trafic VoIP, alors que les paquets jetés par CoDeS doivent être obligatoirement retransmis, utilisant donc de la capacité supplémentaire. La conclusion de ces premières analyses est que CoDeS n'apporte pas d'avantages dans ce cas précis par rapport à PF.

La bonne performance de RR pour un faible nombre d'utilisateurs (moins de 40) peut s'expliquer quant à elle par l'espacement dans l'envoi des paquets d'un même flot sur le canal LMS. Pour un même flot, cet espacement entre les paquets permet de limiter l'impact de retransmissions supplémentaires par HARQ. En effet, un tel paquet ne sera pas détecté comme perdu par TCP car celui-ci n'aura pas le temps de générer trois DUPACK, contrairement à PF, qui a un caractère plus en rafales. Cela permet aussi d'en limiter le nombre de retransmissions. Un exemple de l'impact de RR est donné en Figure 6.24 (page 110).

L'équation (6.9, page 109) montre qu'au-delà de 19 utilisateurs, l'espacement des envois permet d'éviter une retransmission par DUPACK lorsqu'un paquet est retardé de 20 ms (une retransmission supplémentaire par HARQ). Au-delà de ce nombre d'utilisateurs, les performances de RR avec et sans mécanisme de réordonnancement sont similaires, comme montré en Figure 6.25 (page 110). Toutefois, pour un grand nombre d'utilisateurs, l'optimisation plus efficace du canal LMS par PF lui permet d'obtenir des meilleures performances que RR.

Dans un deuxième temps, nous nous intéressons au cas dans lequel le mécanisme de réordonnancement est activé. Les résultats obtenus sont donnés en Figure 1.15.

Nous observons désormais que PF est le seul ordonnanceur à offrir un gain de débit utile plus important que les autres, par rapport à DT_H. RR n'est désormais

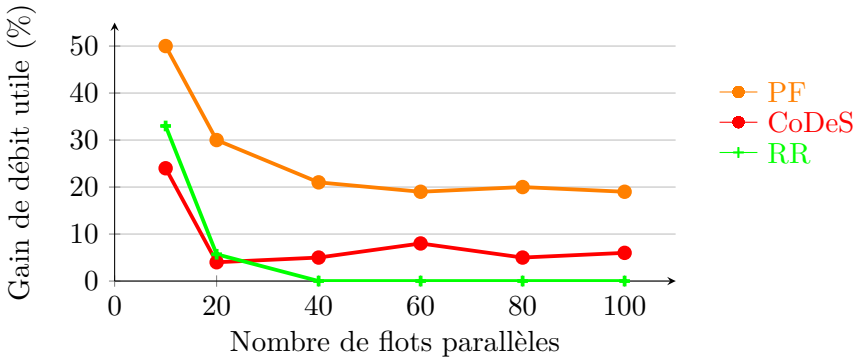


FIGURE 1.15 – Gain de débit utile par utilisateur pour différentes politiques, par rapport à DT_H (CUBIC, réordonnancement)

plus compétitif. En effet, l'avantage apporté précédemment par RR (qui est de minimiser l'impact des paquets désordonnés) devient caduque ici, et la non optimisation du canal LMS ne rend plus RR performant par rapport à des ordonnanceurs optimisant la capacité du canal LMS.

PF reste désormais le seul ordonnanceur à maximiser le débit utile des utilisateurs, de par l'optimisation de la capacité du canal LMS. Il offre un délai de transmission raisonnable, et la majorité des pertes de paquets sont désormais dues à des débordements de files, mécanisme pour lequel TCP sait réagir correctement.

Un autre point à relever est qu'à partir d'une certaine charge dans le réseau, les débits deviennent similaires, pour une même politique d'ordonnancement, et ce quelle que soit la variante de TCP utilisée et de l'activation ou non du réordonnancement. Le gain de performance est alors seulement dû à la politique d'ordonnancement. En revanche, à faible charge, on observe une meilleure performance de CUBIC, et des meilleurs débits avec le mécanisme d'ordonnancement.

Un dernier point à étudier est l'équité entre les flots. En effet, même si les débits moyens sont meilleurs avec PF, nous devons nous assurer que tous les flots aient un débit correct et qu'aucun ne soit grandement désavantagé. Les résultats détaillés, montrant le débit du flot le plus faible et du plus fort, ainsi que l'écart-type sont donnés en Figure 6.4 (page 114) sans réordonnancement et en Figure 6.5 (page 115) avec le réordonnancement.

Nous remarquons que l'écart-type est plus important avec PF qu'avec les autres politiques. Toutefois cet ordonnanceur assure à tous les flots des débits utiles supérieurs aux débits les plus importants avec les autres ordonnanceurs, assurant à tous les utilisateurs un gain de performance avec PF.

L'étude de l'ajout d'un mécanisme d'ordonnancement entre les flots a permis de proposer des solutions pour augmenter la Qualité d'Expérience des utilisateurs. Dans le cas de trafic VoIP, nous avons introduit une nouvelle politique nommée CoDeS, et dans le cas de trafic TCP, nous avons montré la bonne performance d'un ordonnanceur *Proportional Fairness*.

1.5 Conclusion

Nous avons tout le long de ce document mis en avant la nécessité d'étudier l'impact des mécanismes de fiabilisation des couches basses sur les performances des protocoles de transport. Dans notre contexte de constellation LEO, les mécanismes de fiabilisation comme HARQ délivrent les paquets dans le désordre aux couches hautes, engendrant une détection de perte abusive par TCP.

Dans un premier temps, nous avons proposé un mécanisme pour minimiser l'impact du désordonnancement au niveau de la couche transport (TCP), permettant ainsi un gain significatif de débit pour l'utilisateur.

Nous avons ensuite étudié en détail l'impact d'une option de TCP, appelée DelAck, qui modifie significativement les performances de TCP, en bien ou en mal en fonction du scénario. Nous avons montré que l'activation de cette option est contre-productive pour des protocoles de transport agressifs, tels que CUBIC ou TCP Hybla, et que cette option devrait être désactivée dans ces cas-là. De plus, en présence du mécanisme de réordonnancement, DelAck augmente le délai de transmission et la fréquence des timeout TCP, suggérant sa désactivation dans ce contexte.

Enfin en dernière partie, nous avons étudié l'impact de l'ajout de politiques d'ordonnancement sur le canal LMS, afin d'optimiser la capacité du lien. Nous avons introduit une nouvelle politique, appelée CoDeS, permettant d'améliorer la QoE des utilisateurs pour du trafic VoIP. Nous avons également montré qu'un ordonnanceur PF permettait d'optimiser les performances dans le cas de trafic TCP, améliorant la satisfaction des utilisateurs.

Toutes ces conclusions permettent d'améliorer la satisfaction des utilisateurs, que ce soit en termes de débit pour du trafic best-effort, ou de QoE pour du trafic VoIP. Cette étude permet également de mieux comprendre l'interaction entre les protocoles de transport et les mécanismes de fiabilisation du canal LMS.

Chapter 2

Introduction

2.1 Context

Global access to the Internet is not yet a reality. In some cases, such as isolated or moving areas, terrestrial solutions may be very expensive, difficult or totally impossible to deploy. SATCOM allow to overcome Internet access inequalities, providing broader coverage. However, such a solution brings new impairments that need to be deeply studied and tackled. In this context, Thales Alenia Space and CNES lead several studies, concerning satellite capacity management or aiming to reduce latency that can improve the efficiency of satellite transmissions.

In order to minimize the transfer delay and make the use of the terrestrial transport protocols possible without splitting the connection and deploying specific acceleration schemes, the use of **Low Earth Orbit (LEO)** satellite constellations is unavoidable. This kind of constellations allows to connect ground areas with an acceptable transmission delay, similar to terrestrial transmission delays. However, the satellites movements make the links delays dynamic and imply routes changes during a transmission. Moreover, the transmission from a satellite to a mobile ground receiver is difficult, because of the atmosphere, the receiver environment and possible interference. Potential errors on this link need to be efficiently recovered.

To counteract the high error rate, reliability mechanisms have been introduced in the lower layers, aiming to transmit packets on a link as fast as possible while optimizing the capacity. Although such schemes allow to improve the efficiency in the lower layers, their interaction with the transport layer, particularly TCP, needs to be further studied. Indeed, the end nodes are not aware of the presence of a space segment along their network path, leading to the use of standard transport protocols. These protocols are designed for terrestrial networks, where the transmission delay is low and the error rate very low.

This thesis investigates the impact of satellite constellations and their associated reliability schemes on the transport layer performance. The objective is to put forward and explain the bad performance of TCP over LEO satellite constellations. To overcome this issue, new solutions are introduced, deployed on the lower layers,

allowing to prevent any modification at the transport layer that would be hard and costly to deploy. In a second step, optimization mechanisms at the lower layers are analyzed, allowing to optimize the efficiency of the reliability schemes, and thus, the overall performance of the transmissions.

2.2 Contributions and organization

We investigate the impact of LEO satellite constellations on transport protocols, particularly TCP, and bring solutions to improve the global transmission performance. The satellite environment and the reliability schemes are presented in Chapter 3: we present the satellite environment as well as the transport protocols used on terrestrial transmissions and some solutions already proposed for space transmissions.

The major contribution of this thesis is to introduce a new scheduling policy aiming to optimize the efficiency of the lower layers reliability schemes, in the case of several users sharing the network. This policy improves the efficiency of the transmission, and then the user's satisfaction. It is detailed in Chapter 6.

2.2.1 Interaction between the transport layer and the satellite environment

In Chapter 4, we investigate the impact of satellite environment on TCP performance. We drive simulations using Network Simulator 2 (*ns*-2), where we have firstly implemented the link-layer reliability schemes. We then simulate the transmission of a long TCP flow and assess its performance in several scenarios.

We highlight the very low throughput of the transmission, and investigate the cause of this bad performance. Following this first study, we propose a new mechanism working conjointly with link layer reliability mechanisms, aiming to improve TCP performance. We show that this new scheme highly improves TCP goodput, which is the application level throughput, and then overall transmission performance. Finally, the Quality of Experience of the user is highly improved.

We also study the impact of some TCP parameters, such as D-SACK or DelAck on its performance in our scenario, to assess if each one has a significant impact on the throughput. This last study highlights the need to investigate deeper the impact of Delayed Acknowledgment on TCP performance in our scenario. This first work has led to the publication in *International Conference on Wireless and Satellite Systems* [20].

2.2.2 Impact of Delayed Acknowledgment on TCP performance

In Chapter 5, we investigate the impact of Delayed Acknowledgment (DelAck) on TCP performance, always in our context of LEO satellite constellation. This TCP option aims to improve performance on wired networks, and its good performance on terrestrial links explains its default activation in several Operating Systems such as GNU/Linux, MacOS or Windows OS.

The studies, conducted with *ns*-2, showed that the impact of DelAck on TCP performance highly depends on the TCP variant and reliability schemes used: in some cases, DelAck leads to a real improvement of TCP throughput, but can also greatly decrease TCP performance in other scenarios. We analyze the causes of this performance variation to assess when the activation of DelAck is beneficial or detrimental for the transmission performance, and why.

In the cases when the activation of DelAck is beneficial, we investigate what are the better settings of DelAck to have the best performance possible and then optimize the transmission. All this work has been published in *Fifth Federated and Fractionated Satellite Systems Workshop* [21].

2.2.3 Scheduling users to optimize capacity

Finally, in Chapter 6, we study scheduling policies aiming to optimize the efficiency of the reliability schemes, when several users are sharing the associated links. Introducing a good scheduling mechanism can significantly improve the spectral efficiency of the link, allowing to more useful bits to be transmitted and then improving the transmission performance. We test several schedulers and scheduling policies in the different application scenarios corresponding to two different problematics to achieve a good user Quality of Experience: VoIP carried by UDP and best-effort carried by TCP.

The results obtained with these tests lead us to introduce a new scheduling policy named Controlled Delay Scheduler (CoDeS), and show its good performance for VoIP traffic. For best-effort traffic, we show that Proportional Fairness scheduling achieves good performance with a low complexity, as Round Robin in some scenarios. We investigate the impact of the new scheme introduced in Chapter 4, to assess that Proportional Fairness still performs good results with TCP traffic. The results with VoIP are published in *Advanced Satellite Multimedia Systems Conference* [22], and a publication is planned in *International Journal of Satellite Communications and Networking* for the results with TCP.

Chapter 3

State of the Art

Contents

3.1 Context

Reliable transmissions over the Internet allow multiple applications such as web browsing, data transfer or emailing. These transmissions need efficient transport protocol, such as TCP, which is actually the main reliable transport protocol, pervasively deployed and used over the Internet.

Since RFC793 [23], TCP has been constantly improved to follow the Internet evolution: bigger capacity, higher number of users and devices, new applications. TCP remains a protocol designed mainly for terrestrial and wired transmissions, where the transmission delay is short and the link errors are low. In this context, TCP exhibits good performance, adjusting its sending rate to send the most packets as possible without generating congestion collapse, and maintains packet drop rate as low as possible.

However, in some cases, wired transmissions are impossible or difficult to establish, for example:

- the connection of isolated areas to the Internet requires expensive wired connections, which are actually not profitable;
- in the case of moving devices, such as planes, boats, cars or any remote terminal.

As a matter of fact, satellite constellations are a good candidate to provide Internet for these contexts, but at the price of new impairments that do not exist on wired networks, mainly the high link-error rate and the varying transmission delay. Reliability schemes need to be introduced on the space links to mitigate the errors occurring, but their impact on the transport protocols has not been deeply studied.

Particularly, in case of mobile receivers, the link between the satellite and the user can be represented with a **Land Mobile Satellite (LMS) channel** [1, 2, 3], which is characterized by a low average capacity and a high capacity variation. Reliability schemes introduced on this link allow to mitigate the high error rate, but a the price of an increase of the latency and jitter, that have an impact on the transport protocols.

As space links are just a part on the whole network, the use of standard TCP protocols is usual, ignoring the presence of a satellite constellation, but with possible bad performance if no mechanism is introduced to conjointly use TCP, the satellite constellation and the reliability schemes efficiently.

In this chapter, we first describe the satellite environment, with its advantages and weaknesses, then present the reliability mechanisms used to improve the performance of transmissions on the lower layers, especially on the LMS channel. Finally, we present the weaknesses of the TCP variants in this context, and some solutions proposed, suited to this environment.

3.2 Satellite environment

3.2.1 Introduction

The number of satellites is constantly increasing, being good solutions for various applications such as earth observation, military purposes, or telecommunications. Telecommunication satellites enable several applications such as telephone, television, or Internet access. Lower costs and better performance have led to the deployment of more than 2000 communication satellites. Indeed, telecommunication satellites can bring out benefits compared to terrestrial transmissions:

- they can reach any area on the ground, where a terrestrial solution can be difficult or impossible to implement. Thus, they can provide Internet access to isolated areas or mobile devices;
- each satellite covers a large area on the ground. This area depends mainly on the satellite altitude. The more distant the satellite is, the larger area can be covered;
- they can provide a high communication bandwidth. Average bandwidths are around 50 Mb/s, and tends to increase with new generations of satellites, that are more efficient.

The satellite altitude has an impact on several metrics: the transmission delay, which is dependent on the distance between the satellite and the ground, the ground area covered by the satellite, or the power needed to transmit the data. This directly impacts the applications carried by the satellites. Depending on the satellite altitude, telecommunication satellites can be classed into three main categories:

- **GEO satellites**: a unique satellite is used at the Geostationary Earth Orbit and covers a large area on earth. The revolution period being exactly one day, the satellite does not move in the terrestrial reference frame. However, the altitude of the satellite (36000 km) makes the transmission delays important, with $RTT \approx 500$ ms. This is not a problem for applications not needing

interaction such as television broadcast, but makes the use of telephone or Internet protocols very difficult;

- **MEO satellites**: the Medium Earth Orbit (MEO) covers the orbits with an altitude from 2000 km to the GEO orbit. Most of the satellites are at an altitude allowing to be synchronous with earth rotation: 6 or 12 hours for example. Due to the high revolution period, MEO satellites are visible during a large time period, and have a transmission delay lower than GEO, but still not suited for Internet or telephony;

- **LEO constellations**: being at an altitude lower than 2000 km, each LEO satellite covers a small area on ground, and need to be in a constellation to ensure a global coverage of earth. These constellations allow delays close to terrestrial delays, and then the use of the terrestrial transport protocols. However, the low altitude mitigates the coverage for each satellite, and a constellation of several satellites is needed to ensure a global coverage of earth. Moreover, the satellites movements make the routing of the messages more difficult than on terrestrial or GEO networks, and the high speeds force the Doppler Effect to be evaluated and compensated in the transmissions.

For transmissions between two points on earth via one or several LEO satellites, the use of the standard Internet protocols is unavoidable, as the end points ignore the presence of space links. In this context, LEO satellite constellations are adapted to the TCP/IP stack, due to the low delays, comparable to terrestrial ones, and the low attenuation between the satellite and the ground due to the low altitude. This leads to smaller power needed by the satellites and the terminals, and smaller antennas [24]. Then, the low launch cost of a satellite, which is cheaper and more efficient, makes the deployment of a constellation feasible. However, LEO satellite constellations bring new problems specific to this topology that are presented in the next paragraphs.

3.2.2 Topology of LEO satellite constellations

A LEO satellite constellation is often composed of several synchronized satellites (can be up to hundreds of satellites) orbiting around the earth, ensuring a global overage of earth at any time, as in Figure 3.1 obtained with the *SaVi* simulator [25], representing the coverage on the Iridium constellation. On this Figure, we can observe the position of every satellite in green, and its coverage on ground, represented by a yellow circle.

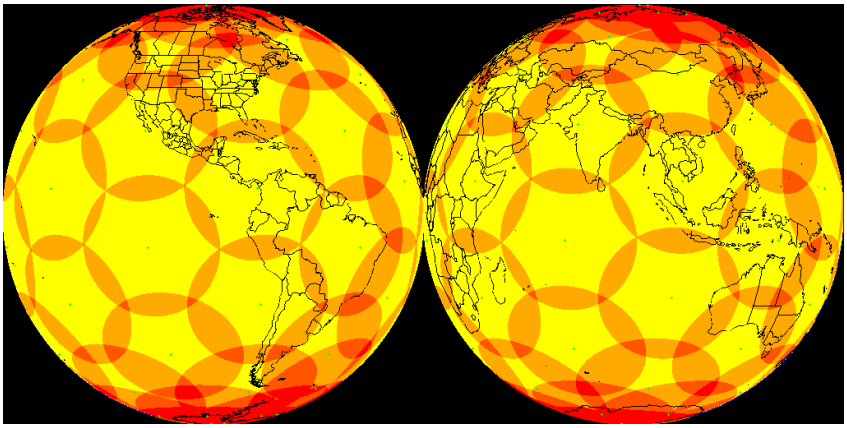


Figure 3.1 – Coverage of Iridium constellation

To ensure this coverage, the satellites are equally distributed on several p planes, each plane having k satellites distributed among the orbit. Thus, the total number of satellites n is such as $n = p * k$. Then to describe a satellite constellation, we need:

- the altitude of the satellites;
- the inclination of the orbital planes (angle between the plane and the poles);
- the number of planes;
- the number of satellites per plane;
- the phase offset between the planes.

Table 3.1 describes some LEO satellite constellations already deployed or being deployed.

Name	Number of satellites	Altitude (km)
Iridium [26]	66	850
Iridium NEXT [26]	66	780
OneWeb [27]	720	1228
ORBCOMM OG2 [28]	18	750

Table 3.1 – Examples of LEO satellite constellations

Moreover, to ensure a better performance of the constellation, constellations can use **Inter-Satellite Links (ISL)** [29] to allow communications between a satellite and its closest neighbors. In this case, a satellite can transfer data to the previous

and next satellites on a same plane and to the closest satellites on the neighbors planes, creating a global grid around the earth, as shown in Figure 3.2.

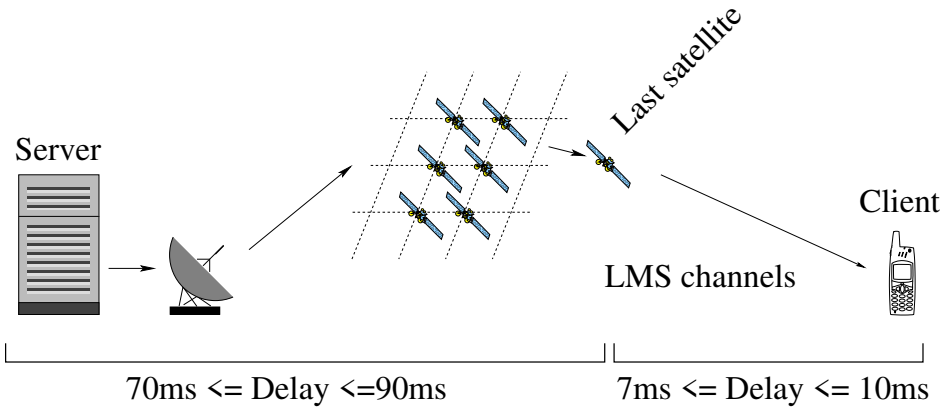


Figure 3.2 – Communication via a satellite constellation with ISL

Today the interest for satellite constellations is growing and new constellations offering bigger capacity are being deployed [26, 27]. However, to ensure a good performance of the system, the protocols need to remain adapted to this environment different from terrestrial transmissions. Internet protocols need to be deeply studied and either optimized or improved to make them adapted for such environment.

3.2.3 Impact of satellite environment on network

LEO satellite constellations bring two new impairments compared to wired networks: varying delays and a high error rate.

Impact of the constellation on the delay: in the context of LEO satellite constellations, the transmission delay, linked to the length of the route, varies between 70 ms and 90 ms, and may have different causes:

- the satellite movements: the route length of a packet may increase or decrease because of the satellite movements. Thus, this change is linear with time, which is not negligible due to the high satellite speed, for example an satellite orbiting at an altitude of 800 km has a speed of 7.5 km/s;

- small handoffs when locally, the route change, one hop switching from one satellite to another, due to a shorter path available. This creates a drop in the total delay of a few milliseconds;

- large handoffs: the packet route totally changes, inducing a sudden large delay variation. This case is very unlikely.

An example of delay variation is given in Figure 3.3, showing the delay evolution between different ground terminals, and the three patterns can be observed. This figure has been obtained with an *ns*-2 simulation, simulating transmissions with a low charge between two points on earth via a satellite constellation.

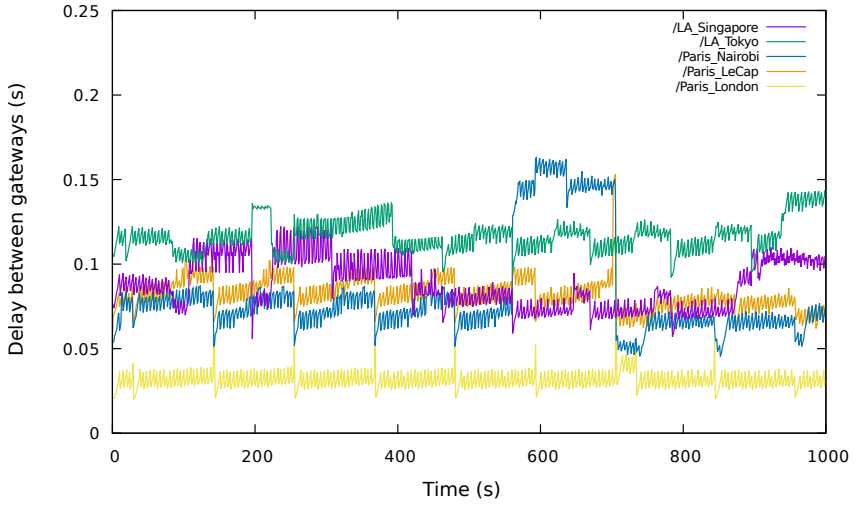


Figure 3.3 – Packet delay evolution for different start and end points

Impact of the constellation on the error rate: another main disadvantage of satellite communications is the high error rate between the last satellite on the message path and the ground receiver. Thus, the travel through the atmosphere, the receiver environment and possible interference generate message errors, which need reliability mechanisms to be recovered. In the case of a mobile receiver on the ground, we can represent this link with a **Land Mobile Satellite (LMS) channel** [1, 2, 3]. The LMS channel is the bottleneck of the network due to its error rate making its capacity lower than the other links, and where the most attention is needed.

The LMS channel cannot be easily represented, this channel is very different from the links between satellites and the links on the ground: the elevation angles are within a large range of values, and is highly correlated with the position of the ground receiver. The channels can be grouped in different categories depending on the receiver location and movements: urban, suburban, rural or open for example. In this thesis, we use attenuation time series provided by CNES, which follow the model presented in [3].

This model of the LMS channel uses a three state Markov chain [1], representing shadowing conditions:

- state 1: line of sight events;
- state 2: moderate shadowing;
- state 3: high shadowing.

In each state the channel can change with different speeds:

- very slow variations, caused by obstacles such as tree or buildings;
- slow variations, caused by the non-uniformity of the obstacles, for example induced by dense vegetation;
- quick variations, caused by multipath.

To cope with this high and varying error rate, reliability mechanisms have been introduced on the lower layers, on the LMS channel, and are presented in Section 3.3.

We can consider the other links error-free, because the atmosphere is not crossed in ISLs, and the errors on the link between the sender and the first satellite can be mitigated with a higher power in the sending terminal. On the return path, errors can be avoided using FEC on the acknowledgments path.

3.2.4 Applications provided by satellite constellations

We now present the different applications that can be provided by satellite constellations. These applications also exist on ground transmissions but the satellite constellation topology allows the use of these applications in a space context. Thus, applications that a LEO satellite constellation can provide are:

- **Data transfer**: this kind of application carries big amounts of data, with a bursty distribution. It is often associated to a best-effort service. All the packets sent must be correctly received, which leads to the use of error-correction or retransmission algorithms. The network must be able to carry the most of data as possible without congestion, which we measure through the goodput metric, corresponding to the quantity of useful bits received within a time window. The main transport protocol used for this service is TCP;

- **Voice over IP (VoIP)**: this application ensures the transport of the voice over wired or wireless networks. This kind of transmissions can accept small losses, but is constrained by the transmission delay and even more by the jitter. The amount of data carried by this kind of application is low, but the delay and jitter constraints may make the transmission difficult, leading to a need to optimize the **Quality of Experience (QoE)** of the users, representing the user satisfaction. As the application does not need to be reliable here, UDP transport protocol can be used here;

- **Live streaming**: this application is similar to the VoIP, but with higher amounts of data. As for the VoIP, this service can handle small losses, but the jitter needs to be minimized. The main transport protocol is also UDP with this service.

Services provided by UDP are used for applications where losses are tolerated but which need a constant transmission time, such as VoIP or live streaming. In this case the useful metrics to measure are:

- **the transmission delay**, which should not be higher than 400 ms [19]. This should not be a problem for LEO satellite constellations, where the transmission delay is approximately 100 ms;

- **the jitter**, *i.e.* the transmission delay variation, which needs to be lowered. This metric is more critical in satellite transmissions, as described later;

- **the losses**. Some losses are tolerated, but within a limit, often around 3 %.

The user satisfaction is very important in these scenarios, and to represent the Quality of Experience (QoE) of the users, a metric called the **Mean Opinion Score (MOS)** [30] has been introduced. Basically, the MOS is computed as follows [30], first with the R-factor:

$$R = R_0 - I_s - I_d - I_e + A \tag{3.1}$$

where:

- R_0 is the basic **Signal Noise Ratio (SNR)**;

- I_s is the simultaneous impairment factor;

- I_d is the delay impairment factor;

- I_e is the equipment impairment factor;
- A is the advantage factor. It is linked to users who can accept a lower quality considering the context of the transmission. With the use of satellite communications, this factor is set at 20 [30].

Then the MOS is computed from the R-factor following Equation (3.2):

$$MOS = 1 + 0.035R + 7.10^{-6}R(R - 60)(100 - R) \tag{3.2}$$

We get a value between 1 and 5, representing the QoE of the users. The majority of users are satisfied when the MOS reaches 3.5.

For video applications, dedicated metrics allow to evaluate the quality of the transmission have been introduced such as **Peak Signal-to-Noise Ratio (PSNR)** or **Structural Similarity (SSIM)** [31]. An adaptation of the MOS has also been made for these applications. However, they will not be presented here, as video transmissions are not in the scope of this thesis.

3.3 Reliability mechanisms

The objective of this section is to present the reliability mechanisms used on the LMS channel to cope with the high error rate. Such mechanisms need to recover the data as fast as possible while optimizing the channel capacity.

3.3.1 Forward Error Correction

Forward Error Correction (FEC), also known as **Channel Coding** is a technique used to ensure reliability on noisy channels. The message is encoded with redundancy, using **Error Correcting Codes (ECC)**. Thus, if a message contains N_k useful bits, the message is encoded by adding N_r redundancy bits. The total size is the $N = N_k + N_r$, and the receiver needs to receive only N_k bits over the N sent to be able de decode the message.

Traditionally, ECC can be classed into two separate categories:

- block codes, used for the protection of data blocks that are independent from the others;
- convolutional codes, used for the protection of data streams.

Several ECC have been proposed to optimize the code-rate: Turbo-codes [32], Low Density Parity Check [33, 34], or Fountain codes [35, 36].

This mechanism minimizes the transmission delay because no retransmission is needed on the lower layers. However, FEC does not optimize the link capacity: in order to handle the worst case, the number of redundancy bits needs to be important, and could use link capacity uselessly when the channel conditions improve. Thus, FEC alone is not suited for LMS channels, as the channel capacity has an important variation over time.

3.3.2 Automatic Repeat reQuest

Automatic Repeat reQuest (ARQ) is a mechanism retransmitting partially or totally lost messages in the lower layers. ARQ is based on an acknowledgment path to inform the sender if a message has been correctly received or not. Several versions of ARQ exist, and are summarized in Table 3.2.

ARQ is used for applications with small error rates, in order to avoid an important number of retransmissions, which can be counterproductive because each packet has a cost in terms of capacity to be transmitted. In the case of satellite communications, the high error rate and the long retransmission delays (20 ms) make the use of ARQ alone impossible.

3.3.3 Hybrid Automatic Repeat reQuest

To take into account the high error rate happening on satellite links, an improvement of ARQ called Hybrid Automatic Repeat reQuest (HARQ) has been introduced, combining FEC and ARQ. Thus, this mechanism allows to optimize the bandwidth of the channel by avoiding too many retransmissions and sending messages without too many redundancy at once. When a packet needs to be transmitted, HARQ first sends the useful bits with some redundancy bits. The receiver side computes if the message can be decoded and if not, asks the sender to retransmit.

Depending on the nature of transmissions and retransmissions, three different versions of HARQ exist:

- **Type I HARQ, or Chase Combining**: while a packet cannot be decoded, the sender sends the same coded message again, and the receiver combines the copies of the message until it is able to decode the packet, as shown in Figure 3.4. This version does not optimize the channel capacity, but needs a few memory capacity at the receiver;
- **Type II HARQ, or Incremental Redundancy**: in this version the first sending contains the useful bits and some redundancy bits. Then each time

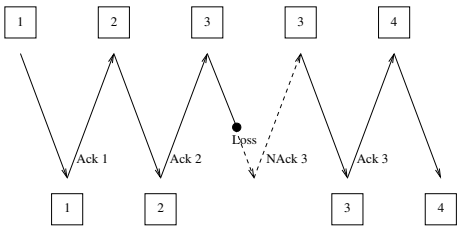
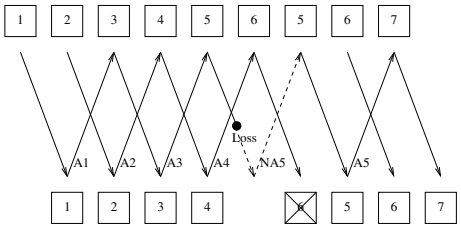
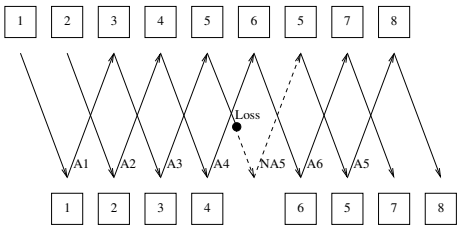
ARQ version	Illustration
Stop and Wait: this is the simplest version of ARQ: the sender sends a packet to the receiver, and waits for the acknowledgment or a timeout to to send the next packet or retransmit the same packet. This waiting time makes this version counterproductive, the channel capacity being not used, particularly on a satellite link where the RTT is around 20 ms	 Type I ARQ
Go-back-N: this is an improvement of Stop and Wait ARQ, where the sender does not wait an acknowledgment to send the next packets, within the limits of a fixed time window. Thus this version improves the usage of the bandwidth of the channel. However, if a packet is not acknowledged, the sender will retransmit from the packet lost, and send again the next packets that may have been correctly received (this is called a spurious retransmission)	 Type II ARQ
Selective Repeat: this version is designed to have a better handle of out-of-order packets. The sender behaves like in the Go-back-N ARQ, and does not waits for an acknowledgment to send the next packets. On the other side, when a packet is missing, the receiver sends a negative acknowledgment to the sender, but stores the next packets, waiting for the retransmission of the packet lost. This needs to have a buffer to store the packets when a loss is detected, but it improves the performance of ARQ compared to the previous versions	 Type III ARQ

Table 3.2 – The three ARQ schemes

a retransmission is asked by the receiver, the sender sends new redundancy bits, in order to increment the total number of redundancy bits received. An example is shown in Figure 3.5. This version is more efficient than Chase Combining, by offering more redundancy bits at each retransmission;

- **Type III HARQ**: in this version, at each retransmission, the packet sent is totally different from the packets from previous sending, and can be decoded itself alone.

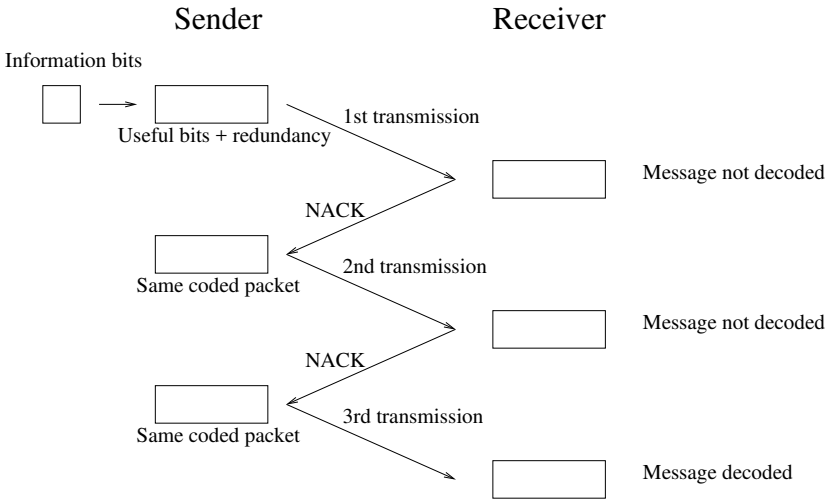


Figure 3.4 – Type I HARQ

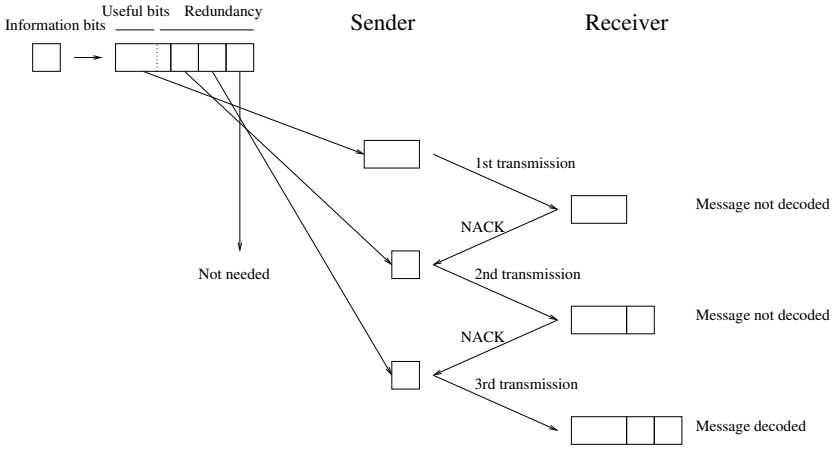


Figure 3.5 – Type II HARQ

Some improvements have been proposed to optimize these algorithms. Thus we can quote an improvement of Type II HARQ, called Adaptive-HARQ [4], which optimizes the LMS channel usage by using Mutual Information to compute if a

message can be decoded or not, as described in Figure 3.6. This scheme allows up to three retransmissions, and can handle most of the cases while optimizing the transmission time and the channel usage. This version shows good performance in terms of channel optimization, explaining our choice to use this one in this thesis.

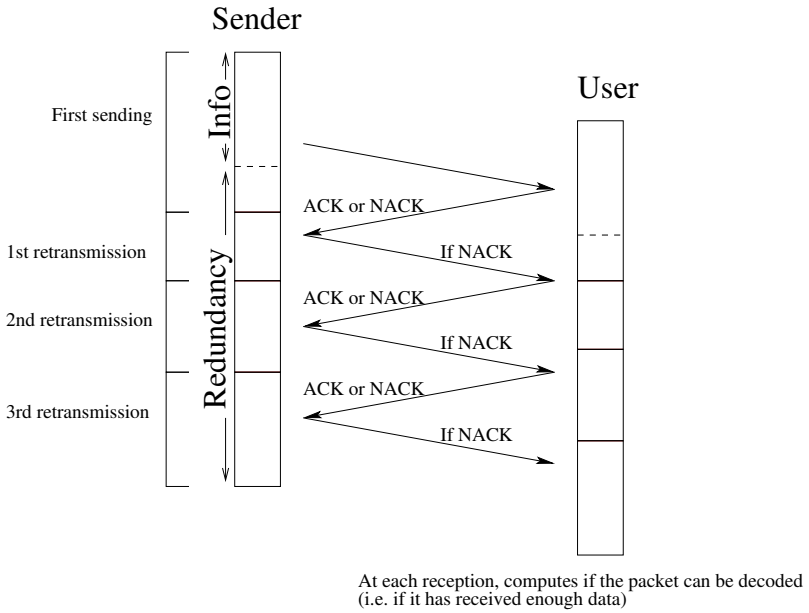


Figure 3.6 – Description of Adaptive-HARQ

3.3.4 Applications of HARQ

HARQ is mainly used in terrestrial wireless networks. We can cite **High Speed Downlink Packet Access (HSDPA)** or **High Speed Uplink Packet Access (HSUPA)** protocols, providing high speed transmission for mobile telecommunications networks. The **IEEE 802.16-2005** standard, also known as "mobile WiMAX" uses HARQ, as well as **Evolution-Data Optimized (Ev-DO)** and **Long Term Evolution (LTE)** wireless networks.

Compared to terrestrial wireless networks, the satellite links have a bigger propagation delay and a different channel model. This leads to a different tuning of HARQ parameters in order to take these differences into account.

3.4 Adapting the transport layer to the satellite context

Transmission Control Protocol (TCP) has become the main protocol in the transport layer, showing good performance on terrestrial networks. It is mainly driven to react and adapt to congestion that can happen in the network. However, satellite communications and associated reliability schemes bring new issues that have to be considered, such as longer transfer delays and higher jitter.

One solution could be to isolate the space segment by introducing **Performance Enhanced Proxies** (PEP) [37, 38], and using more adapted transport protocols on this sub-network [39]. However, such a solution has some weaknesses, for example the cost of this kind of mechanisms, as well as End-to-End security issues [40]. Moreover, PEPs are designed to split the connection in long delays networks or to isolate the space segment from the terrestrial segment. In our context, we have reasonable delays and our goal is to analyze the impact of reliability schemes on the whole connection. Thus, PEPs are not an optimal solution in our context, and it could be more efficient to use a non splitted connection, with the terrestrial protocols, at the price of some schemes introduced on the lower layers to improve overall performance.

In the next paragraphs, we present the main weaknesses of TCP in the case of LEO satellite constellations and the solutions proposed.

3.4.1 TCP weaknesses in LEO satellite constellations

A LEO satellite constellation can be considered as a **Long Fat Network (LFN)**, as its Bandwidth-delay product is higher than 10^5 bits [41]. This high product has a strong impact TCP performance for three main reasons:

- the maximum window size, which is 65535 bytes, can be reached. This issue has been fixed [42], allowing higher windows, and is now enabled in the main Operating Systems;

- the cumulative acknowledgment policy. When a packet is detected lost, TCP has no information about the good reception of the next ones, and this may lead to the unnecessary retransmission of these packets, decreasing TCP performance. This problem has been fixed with the introduction of **Selective Acknowledgment (SACK)**, informing the sender on the packets that really need to be retransmitted;

- the RTT is difficult to estimate, leading to a worse performance of TCP timeout. With this poor RTT estimation, TCP cannot react correctly to changing traffic conditions, especially when packet losses occur. The TCP Timestamp option has been introduced to counteract this problem allowing a better approximation of the RTT [43].

Another main issue for LEO satellite constellations is the high error rate. Indeed, TCP throughput is driven by two parameters that are the Round Trip Time RTT and a loss probability p, and follows approximately equation 3.3 [44]:

$$TCP_{throughput} \approx \sqrt{\frac{3}{4}} \frac{MSS}{RTT\sqrt{p}} \tag{3.3}$$

We can see that TCP throughput is maximized when the RTT and the losses are low, which is not the case in satellite transmissions. Even though the RTT is still in an order of magnitude allowing the use of TCP, *i.e.* between 70 ms and 100 ms, the main disadvantage of using satellite transmissions is the high error rate. The use of reliability schemes as presented in section 3.3 allows to decrease the number of losses but at a price of a higher transmission delay and jitter.

Furthermore, TCP always considers losses as congestion events, and decreases the congestion window to prevent congestion collapse. However, in satellite transmissions, losses are often due to channel impairment, without any correlation with congestion in the network. This leads to a congestion window halved by TCP whereas the sending rate could be more important.

Finally, the changes in the route length, leading to transfer delay variations and out-of-order packets also decrease TCP performance, because of the generation of Duplicate Acknowledgments as it is described later.

All these problems lead to the need to adapt TCP to the context of LEO satellite constellations, by using different protocols or improving the existing ones.

3.4.2 Impact of delay variation on TCP Retransmission Timeout

TCP Retransmission Timeout (RTO) allows to detect a loss at the end of a flow using a timeout. If no Ack has been received at the expiration of the timer, the packet is considered lost and is retransmitted. This timeout value is computed and updated during the transmission following equation 3.6 [45], in order to adapt to the evolution of the RTT and the jitter, and to react as soon as possible to a loss without causing spurious retransmissions.

$$SRTT = (1 - \alpha)SRTT + \alpha R' \tag{3.4}$$

$$RTTVAR = (1 - \beta)RTTVAR + \beta|SRTT - R'| \tag{3.5}$$

$$RTO = SRTT + 4RTTVAR \tag{3.6}$$

In this equation:

- R' is the latest RTT measurement
- $SRTT$ (Smoothed RTT) is the exponential moving average of RTT
- $RTTVAR$ (RTT Variation) is the exponential moving average of the distance between R' and $SRTT$

Depending on the values of α and β, the evolution of the RTO timer differs. High values of α and β make the timer to follow only the latest measurements of RTT and not take into account the previous measures. In the opposite, way too small values keep out the RTO to follow sudden changes within in the RTT. RFC 6298 recommends to use $\alpha = 1/8$ and $\beta = 1/4$ as a good compromise.

The use of reliability schemes on the LMS channel increases the transfer delay (and so the RTT) and the jitter. Even if the RTO timer value can adapt to these delays changes, it cannot always prevent false loss detection and spurious retransmissions. The impact of the RTO on the transmission performance must be carefully considered.

3.4.3 Impact of out-of-order packets

Section 3.3 described the HARQ reliability mechanism. Depending on the channel quality during the packet transmission on the LMS channel, due to possible retransmissions between the satellite and the ground, the decoding time can vary, causing packets to be delivered out-of-order to the transport layer. Moreover, LMS channel having a high variation of capacity, the number of out-of-order packets is important and cannot be neglected.

A packet p needing one more retransmission than the others will arrive 20 ms later at the receiver, and then potentially after dozens of packets sent after p, as shown in Figure 3.7. On this figure (and the following), the left side corresponds to the sender and the sequence number of the packets sent, while the right side is the receiver with the Acks corresponding to the last received packet. This missing

packet causes the TCP receiver to generate DUPACK, and to trigger a spurious retransmission. Because the TCP sender considers the packet is lost due to congestion, it also halves its congestion window, decreasing its sending rate.

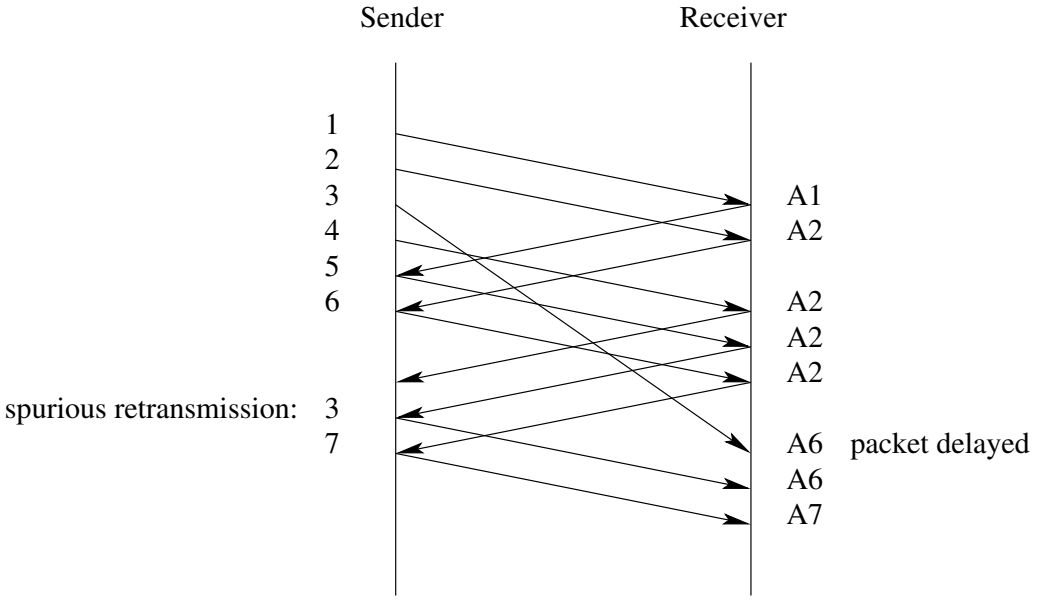


Figure 3.7 – Impact of out-of-order packets on TCP

Out-of-order packets can be also caused by the changing delays in the constellation, which have exactly the same impact than the reliability scheme, but which are less frequent.

The use of the Selective Acknowledgment [46] option allows to mitigate the number of spurious retransmissions by retransmitting only the packets that have been detected as lost. However, SACK does not have an effect on the cause of out-of-order packets.

This points out the need to fully understand the interaction between the reliability schemes and the transport protocol in our context of LEO satellite constellation transmissions.

3.4.4 Impact of Delayed Acknowledgments

In order to improve TCP performance on terrestrial networks, a mechanism called Delayed Acknowledgment [10] (DelAck) has been proposed. The aim of this option is to combine two acknowledgments if the packets have been received in-order within a fixed time window (often 40 ms), as seen in Figure 3.8. If one of the two previous conditions is not achieved, an immediate Ack is directly sent to the sender in order to let it recover a potential error as soon as possible. This TCP option reduces

the feedback path load by lowering the number of Acks, and thus improves the end-to-end performance. DelAck also reduces the CPU load [11], explaining why this option is now mainly activated by default.

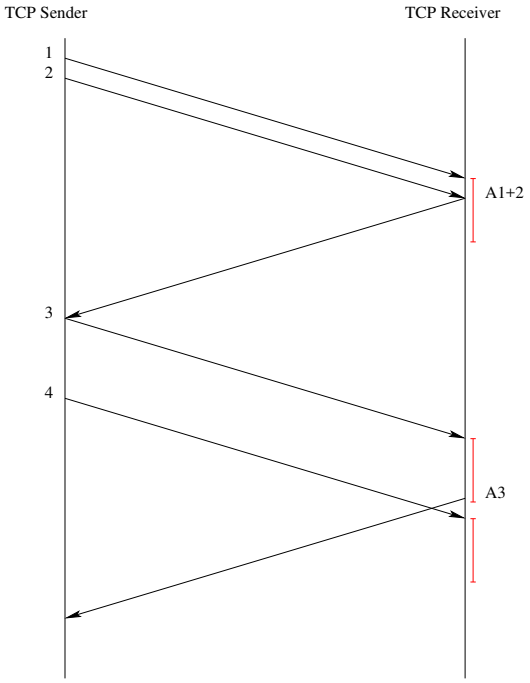


Figure 3.8 – TCP Delayed Acknowledgment mechanism

Finally, the congestion window being updated at each Ack reception, DelAck enables pacing in the transmission [47] as shown in Figure 3.9, obtained with two independent *ns*-2 simulations, each one with a single flow performing between two nodes, one with DelAck and the other without DelAck. DelAck helps the congestion window to find its optimal value in congested networks without saturating it. This also increases TCP performance in networks with small error rates, such as wired networks, but also significantly increases the slow start duration in Slow Start phase.

Some improvements of TCP have been proposed to ensure a better performance with DelAck, such as Appropriate Byte Counting [48], or adaptive DelAck time (implemented in GNU/Linux kernels).

3.4.5 Impact of Slow Start on short-lived flows

Most Internet flow are short-lived, composed of a dozen of packets or less. This low number of packets does not allow TCP to reach its optimal congestion window, which stays in slow start phase. Then the impact of the **Initial Window (IW)** is important on TCP performance. An IW of one packet implies a transmission

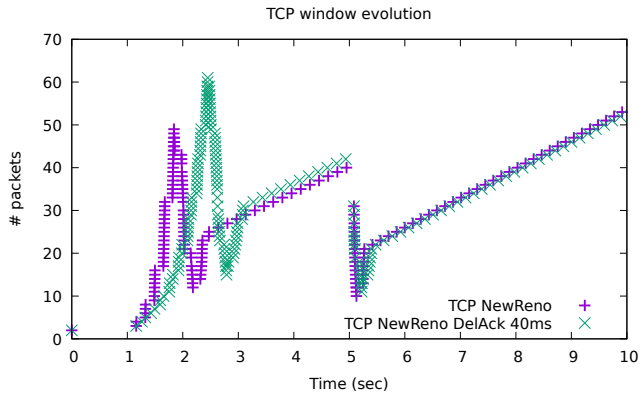


Figure 3.9 – Evolution of TCP congestion window with and without DelAck

time of 3 or 4 RTTs for a short-lived flow of a dozen of packets. To improve the transmission performance, several improvements have been proposed:

- begin with a larger IW. This solution reduces the number of RTT needed to transfer the packets, but creates bursts in the network, causing congestion and buffer overflow, and finally decreasing TCP performance [49]. This solution is already deployed in some real systems;
- pacing has been proposed to reduce the impact of the bursts in the network [47]. The packets are in this case distributed along the whole RTT to let the intermediate nodes enough time to forward the packets without storing them in the buffer. However, it has been shown that such pacing delays congestion signals and synchronizes losses. Finally this option may become counterproductive, as we show in Chapter 5;
- some solutions improve the pacing algorithms and show good performance on short-lived flows, for example Initial Spreading [50]. We can also quote Transmit Twice and Transmit FIN [51] aiming to reduce the number of timeouts and improve the recovery.

In the context of satellite constellations, where RTTs are a bit longer than on terrestrial networks, the improvement of the Slow Start for short-lived flow is essential. However, long flows are not highly concerned by these mechanisms as the Slow Start is negligible compared to the total number of packets transmitted.

3.4.6 Recent Acknowledgments (RACK)

This new loss detection algorithm [52] (the development is still in progress) uses the notion of time instead of duplicate acknowledgments to detect losses. RACK deems

a packet lost if some packet sent sufficiently later has been delivered, as shown in Figure 3.10. This new approach makes RACK adapted to increasingly common patterns observed in the Internet, such as out-of-order packets, tail drops or lost retransmissions.

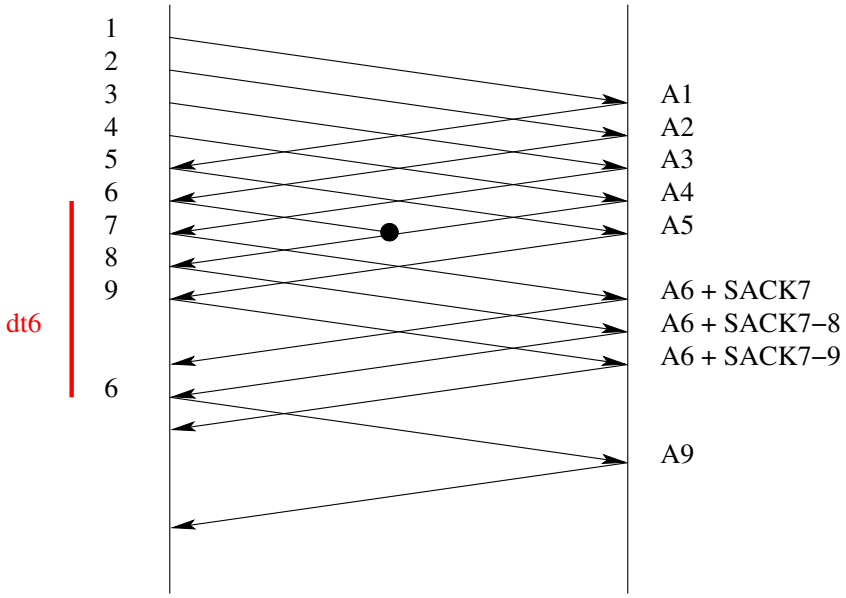


Figure 3.10 – Transmission using RACK

This new loss detection algorithm, which does not need changes on the TCP receiver side, uses the most recently delivered packet's transmission time to judge if packets sent previous to that time have expired. It allows packets to be retransmitted without waiting for a RTO and then resetting the congestion window. It can also replace the fast recovery algorithms.

RACK algorithm uses the following variables:

- Packet.xmit_ts: time of last transmission of data packet;
- RACK.RTT: RTT measured;
- RACK.reo_wnd: reordering window (by default: 1 ms or if reordering is detected: RACK.min_RTT/4).

A packet is considered as lost if the equation (3.7) is true. This equation is computed for all the packets at each Ack reception.

$$now > Packet.xmit\_ts + RACK.reo\_wnd + RACK.RTT \tag{3.7}$$

Figure 3.11 shows a packet that has been delayed in the network. The use of classic loss detection algorithms would induce a generation of DUPACK and then a spurious retransmission of the packet. However, with the RACK approach, there is no loss detection, nor congestion window decrease, because the acknowledgment is received before equation (3.7) is fulfilled.

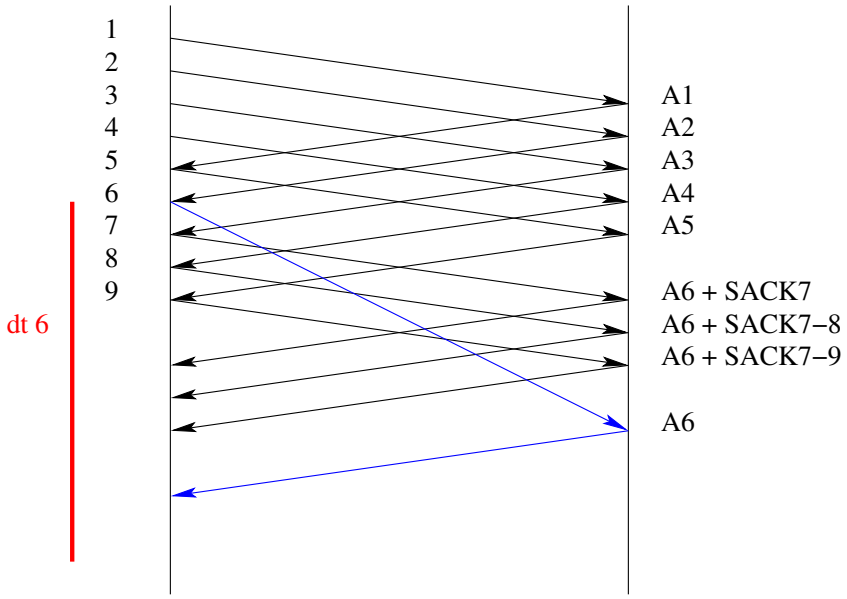


Figure 3.11 – Impact of out-of-order packets with RACK

Moreover, RACK uses also the **Tail Loss Probe (TLP)** [53] mechanism to reduce RTO recoveries. This algorithm allows to detect tail losses without waiting for an RTO and to reset the congestion window. It works as follows:

- defines a Probe TimeOut (PTO), where this value is $max(2 * SRTT, 10ms)$, and is smaller than RTO
- when the timer expires, transmits a loss probe containing:
  - a new segment if available;
  - if not, the most recently sent segment;
- Ack (containing SACK) from probe triggers SACK-based loss recovery, when there is tail loss

In figure 3.12, we can see an example of the use of TLP to detect a tail loss and retransmit lost packets. Note that the SACK option is mandatory to detect the missing packets.

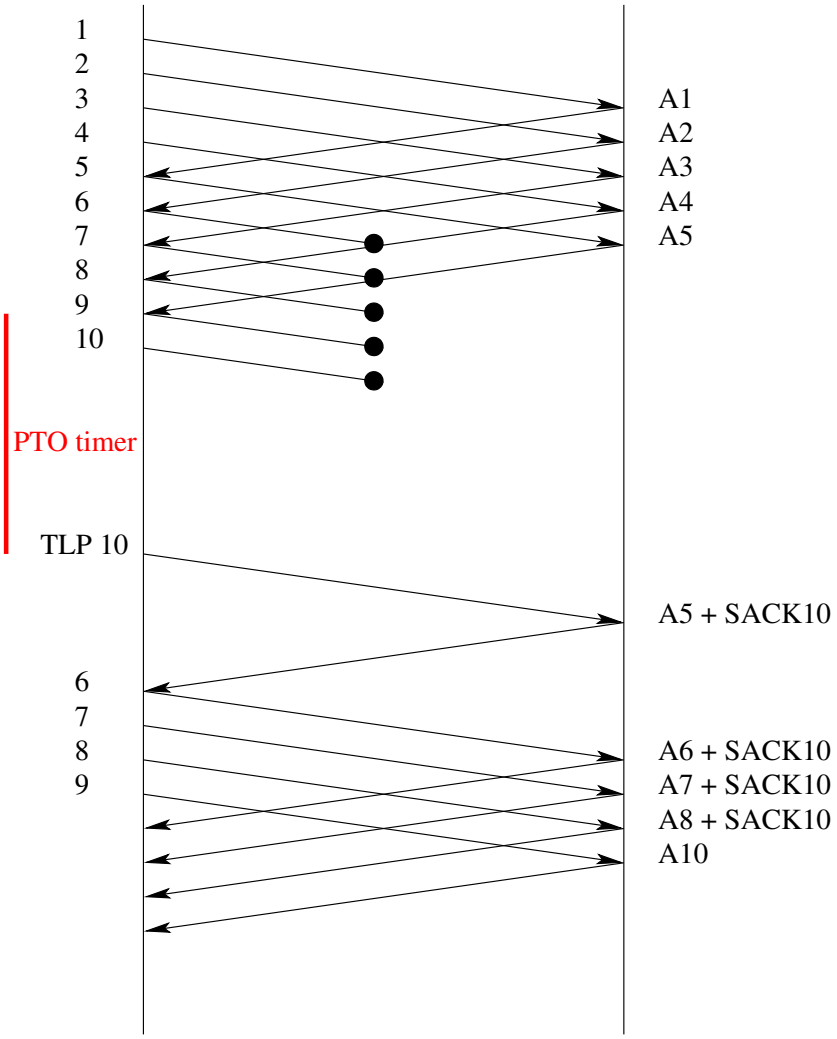


Figure 3.12 – Example of retransmission using Tail Loss Probe

RACK and TLP have shown good performance, and the trend is to let these loss detection algorithms replace DUPACK et RTO algorithms. That is why it already implemented in GNU/Linux since 4.4, and improvements are still expected to improve its performance.

3.4.7 Standard TCP variants

The sender of the transmission is not aware of the presence of a space segment in the network, and then often uses the main TCP variants as transmission protocol. In the next paragraphs, we describe the two main TCP transport protocols: TCP NewReno and CUBIC.

TCP NewReno: TCP NewReno [7] is one of the most simple variants of TCP. We use it in our simulations as in reference for our measures. It is the evolution of TCP Reno and the main algorithms are as follows:

- in Congestion Avoidance phase, at each acknowledgment reception the congestion window is updated following: $cwnd+ = 1/cwnd$. Thus, the window growth is 1 packet per RTT;
- when entering the Fast Retransmit phase, the system is updated as follows: $ssthresh/ = 2$ and $cwnd/ = 2$.

The congestion window evolution is slow with this TCP variant, and some more recent algorithms show better performance, but its simple evolution makes TCP NewReno a good candidate to study and to help understand TCP behavior.

CUBIC: CUBIC [8] is a TCP variant designed for high bandwidth networks with high latency. The name CUBIC comes from its window evolution which is a cubic function. During the Congestion Avoidance phase, the window evolves as follows:

$$cwnd = W_{max} + C(t - K)^3 \tag{3.8}$$

where:

- W_{max} is the maximum window size before the last reduction;
- t is the elapsed time from the last reduction;
- C is a scaling factor;
- $K = \sqrt[3]{W_{max}\beta/C}$ where β is a constant multiplication decrease factor applied for window reduction at the time of loss event.

We note here that CUBIC evolution does not depend on the reception of acknowledgments to update the congestion window, but only on the time elapsed since the last loss, allowing more fairness between CUBIC flows. Following these equations, the window evolution of CUBIC looks like the evolution shown in Figure 3.13, obtained with *ns*-2, with a single flow performing between two nodes.

CUBIC is largely deployed today in the Internet, it is for example the default TCP variant of GNU/Linux, OSX systems (since `10.9`) and Windows (since `Windows 10 Fall Creators Update`)

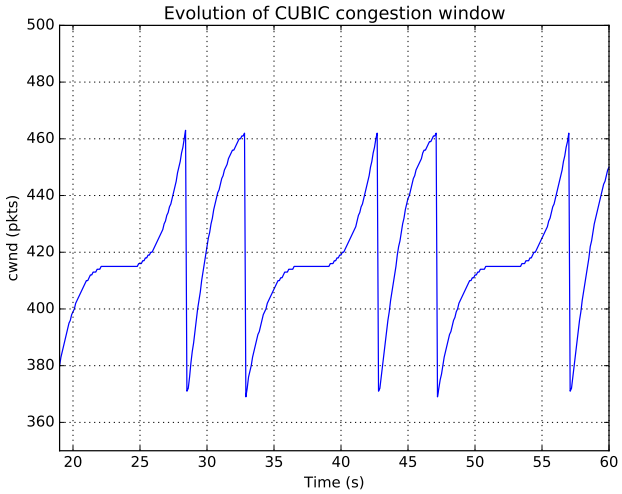


Figure 3.13 – CUBIC congestion window evolution

The predominant position of this variant explains why we study CUBIC on LEO networks. LEO satellite constellation are not using proxies to isolate the spatial segment, thus many LEO networks will carry CUBIC flows, without optimization mechanisms as in PEP [54] architectures.

3.4.8 Other TCP variants for space transmissions

Several TCP variants have been proposed to deal with network impairments. We introduce the protocols that have been proposed to deal with Long Fat Networks and LEO satellite transmissions.

TCP Westwood: TCP Westwood [55] is a TCP variant which makes the difference between congestion losses and link-error losses.

TCP Westwood uses the Acks reception to measure the end-to-end bandwidth. Then when a loss is detected, TCP Westwood uses the measured bandwidth to adapt its congestion window and the slow start threshold, unlike automatically halving the congestion window when a loss is detected due to DUPACKs when other variants are used. This algorithm is called **faster recovery**.

An improvement of this protocol has been proposed call **TCP Westwood+** [56], improving the bandwidth estimation, and is now implemented in the Linux Kernel.

TCP Westwood shows good performance over wireless channels where losses due to channel impairments occur, by differentiating congestion losses from link-error losses. Moreover, this variant shows good fairness with other protocols, and can

be conjointly mixed with other TCP variants, and without the use of proxies at intermediate nodes.

TCP Hybla: this TCP variant has been designed for long RTT links [12], as Geostationary links. As standard protocols are penalized by high RTT in satellite transmissions, TCP Hybla removes the RTT dependence by introducing a normalized Round Trip Time ρ:

$$\rho = \frac{RTT}{RTT_0} \tag{3.9}$$

In equation 3.9, RTT_0 represents the Round Trip Time of a reference connection to which TCP Hybla aims to mimic the performance. The congestion window is then updated as follows:

$$W_{i+1}^H = \begin{cases} W_i^H + 2^\rho - 1 & \text{(SS)} \\ W_i^H + \rho^2 / W_i^H & \text{(CA)} \end{cases} \tag{3.10}$$

Equation (3.10) shows that the transmission rate is independent of the RTT. This leads to higher congestion windows compared to classic TCP variant (TCP NewReno or CUBIC for example), and then to a high error rate. TCP Hybla implements some countermeasures to take into account the number of losses.

TCP Hybla shows good performance over GEO links compared to the other protocols. However, this variant has some drawbacks:

- this protocol does not ensure fairness between the different flows if other TCP variants are sharing the network;
- the lack of fairness requires the satellite links to be isolated from the rest of the network, with Performance Enhanced Proxies (PEP) [38], the satellite part allowing only Hybla flows.

TCP Peach: TCP Peach [57] has been proposed for satellite networks to counteract the effect on high delays and high error rates. It introduces the notions of **Sudden Start** and **Rapid Recovery**, replacing the usual Slow Start and Fast Recovery schemes. It also sends dummy segments in the network to measure its capacity.

Dummy segments are low priority packets containing no data, which are the first packets dropped by the network when congestion appears due to their

low priority. Depending on the number of acknowledged dummy segments, TCP Peach can estimate the network capacity: unused capacity in the network allows the dummy segments to be acknowledged, allowing TCP Peach to increase its sending rate. However, the receiver must be able to recognize dummy segments and to acknowledge them. If not, TCP Peach works as TCP Reno.

The Sudden Start algorithm, replacing the Slow Start, starts with an Initial Window of 1, sends a data packet and then $rwnd - 1$ dummy segments distributed along the first RTT, where $rwnd$ is the maximum value of cwnd. At each reception of an Ack for a dummy segment, TCP interprets this a free capacity and increases its window by 1, then transmits new data segment. At $t \approx RTT$, all dummy segments are sent and TCP Peach switches to Congestion Avoidance phase. Sudden Start shows a quicker congestion window increase than Slow Start, without creating congestion on the data packets due to the low priority of the dummy packets.

The **Rapid Recovery** algorithm replaces the Fast Retransmit algorithm called when a loss has been detected via DUPACKs. As for Sudden Start, Rapid Recovery uses dummy segments to update its congestion window, then switches to Congestion Avoidance phase.

TCP Peach shows good performance over satellite networks without degrading the fairness. However, it needs a slight modification of the end receiver in order to take into account the TOS field and then manage the priority of the dummy segments.

TCP Noordwijk: TCP Noordwijk [58] is a transport protocol designed by the European Space Agency for web traffic on satellite networks. It is optimized for well-known environments such as a **Digital Video Broadcasting - Return Channel via Satellite (DVB-RCS)** link between **Interoperable – Performance Enhancing Proxies (I-PEPs)**.

Instead of being window-based, this variant is wave-based, where a wave corresponds to a burst of packets. The wave is composed of a BURST size, and a time interval during which BURST cannot be changed, and is updated at Ack reception.

TCP Noordwijk optimizes transfer of short files over DAMA access files such as DVB-RCS, and improves performance compared to protocols using the Slow Start phase in high delay-bandwidth environments. Even if designed for transfers of small amounts of data, it also shows good performance for long transfers.

Bottleneck Bandwidth and Round-trip propagation time (BBR): this new protocol [59] does not rely on the packet losses to lower the congestion window, but is model-based. In this variant, the sending rate is only driven by two constraints: *RTprop* (Round Trip propagation Time) and *BtlBw* (Bottleneck Bandwidth), that allow to fully characterize the network. These two variables are constantly updated during the transmission to adapt to networking conditions. This approach allows BBR to separate from the classic TCP rule *loss = congestion*, which was true in the beginning of TCP but not now due to the evolution of the network characteristics: higher throughputs or bufferbloat [60]. Moreover, BBR brings a solution to the RTT estimation issued in Section 3.4.1.

BBR ensures fairness between several BBR flows, while this protocol performs throughputs way higher than CUBIC, according to Google experiments [59]. This had lead the company to begin the deployment of BBR on *google.com* and YouTube video servers, increasing the Quality of Experience of the users.

3.4.9 Explicit Congestion Notification (ECN)

Explicit Congestion Notification [61] is an extension of the Internet Protocol and TCP allowing to notify congestion without dropping a packet.

This scheme only works when **Active Queue Management (AQM)** [62] policy is present. This queue management can mark packets when the average queue size is higher than a threshold. This behavior allows AQM to use other mechanisms than dropping the packet. The queue uses the **Congestion Experienced (CE)** codepoint in a packet IP header to notify congestion. A packet with such a codepoint is called *CE packet*.

ECN allows this notification of congestion without dropping packets, when it is used between two ECN-enabled endpoints. The routers supporting ECN can then fill the CE codepoint of a packet to notify congestion, and let TCP react to this congestion by adjusting its sending rate. This algorithm allows to avoid retransmissions of packets, reducing the latency and the jitter.

TCP supports today ECN by default, using flags in the TCP header. This option needs to be negotiated at the connection establishment, allowing then the intermediate nodes to mark the packets with the CE codepoint instead of dropping them. The trend today is to deploy widely ECN, and the majority of web servers are now able of negotiating ECN [63].

ECN may have applications such as **Congestion Exposure (ConEx)** [64], informing the intermediate nodes of the expected congestion. The intermediate

nodes can then limit a flow throughput if it does not respect the fairness between the flows.

3.4.10 Duplicate SACK (D-SACK)

Duplicate SACK (D-SACK) [65] is an extension of TCP Selective Acknowledgment (SACK) option used to detected the reception of duplicated packets. With D-SACK, the first block of the SACK option is a D-SACK block giving the sequence number of the duplicated packets.

The use of D-SACK can help the sender to determine the origin of this duplicate: replication by network or out-of-order packet retransmitted or Ack lost for example. With this information, the sender can adapt to networking conditions, for example adapt the duplicate acknowledgment threshold [66] if out-of-order packets are detected, to let delayed packets enough time to be received without triggering a Fast Retransmit.

3.5 Conclusion

The use of a LEO satellite constellation induces varying delays and high channel impairments on the LMS links, recovered with reliability mechanisms such as HARQ. These constraints, that are not significant in wired networks, highly decrease TCP performance in a satellite environment.

We have presented here some solutions proposed to overcome these issues. Some of them are TCP options that can help TCP to adapt to the network conditions, some others are new congestion control algorithms specifically designed for these environments. However, the use of LEO satellite constellations, unlike GEO satellites, does not allow to use PEPs to isolate the space segment. Standard TCP protocols have to be used, ignoring the presence of a space segment. Even if they are suited for such environments, because of the reasonable transmission delays, we show in the next chapters that their performance is halved by the varying delays and the reliability schemes on the LMS link. Following this statement, we propose schemes aiming to improve TCP performance.

Chapter 4

Mitigating the impact of out-of-order packets

Contents

4.1 On the need to understand the impact of HARQ on TCP

As we have seen, the use of a LEO satellite constellation brings new impairments to the network which are:

1. a slightly higher transmission delay compared to terrestrial transmissions;
2. a changing delay due to the satellite movements and the route changes;
3. a high link impairment between the last satellite on the route path and the ground receiver, which is a LMS channel is our case.

To counteract the third point, reliability schemes such as HARQ have been introduced on the LMS link, at the price of a higher delay, a higher jitter seen by the transport layer and the reception of out-of-order packets.

Concerning the transport layer, the absence of PEP on a LEO satellite constellation implies the use of the protocol chosen by the end nodes, which are not aware of a satellite network. Then the classic variants such as TCP NewReno and CUBIC are used, the order of magnitude of the delays being compatible with these protocols. However, even if TCP reorders the packets before sending them to the application layer, its loss detection algorithms can be misled by out-of-order packets, halving its performance.

We show in the next sections that TCP performance in this scenario is very low, less than 500 kb/s, while we could expect a goodput of 40 Mb/s on a lossless network. There is then a need to understand the impact of the satellite constellation and HARQ on TCP, and then to find solutions improving the performance of the transmission.

4.2 Scenario

4.2.1 Satellite environment

The simulations are made using Network Simulator 2 (*ns*-2), allowing to simulate LEO satellite constellations. The constellation chosen is composed of 66 LEO satellites, at an altitude of 800 km, allowing a global coverage of any point of Earth at any time. This kind of constellation corresponds to real constellations already or being deployed. The topology simulated is described in Figure 4.1.

To ease the simulations and make them faster, we modeled the satellite constellation using a three nodes network, and changing delays on the links. The first link

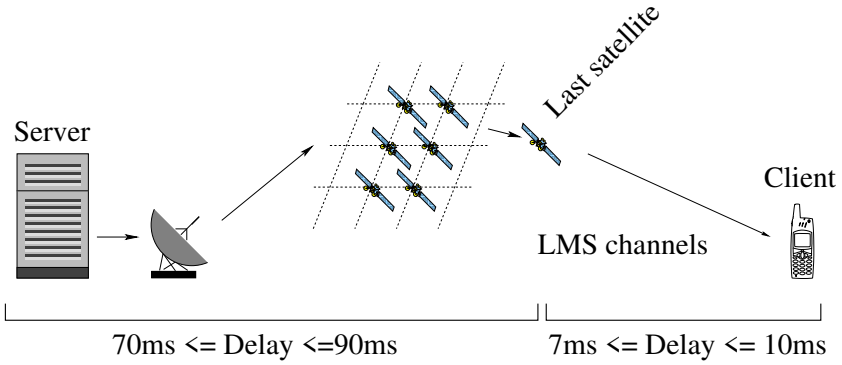


Figure 4.1 – Satellite environment with LMS channel

represents the packet route between the sender and the last satellite, the second represents the link between this satellite and the end user on ground, *i.e.* the LMS link, as shown in Figure 4.2.

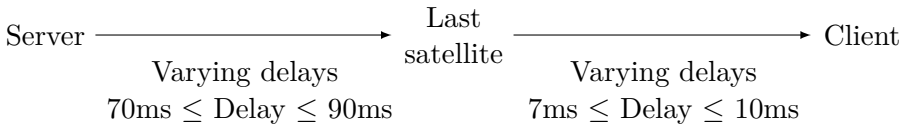


Figure 4.2 – Simplified model used in *ns*-2

The LMS channel model we are using (described in [3]) corresponds to a bandwidth value for this link in *ns*-2 of approximately 50 Mb/s. On the other links, we set a bandwidth high enough to avoid congestion and have the bottleneck on the LMS link. Moreover, we drove other simulation with higher and lower link bandwidth values, and found the same conclusions than with a value of 50 Mb/s. Thus, the value chosen in our simulation is not a particular case.

To get this simplification, we first launched a first *ns*-2 simulation using data from `SaVi` [25] to get the orbit parameters. This simulation, using UDP traffic with low load to avoid congestion, and simulating all the satellites of the constellation, gives temporal traces on the delays between the sender and the last satellite, and between the last satellite and the receiver, corresponding to the LMS channel.

This temporal trace is then used in all the simulations, using the 3 nodes model, to mimic the movements of the satellites. An overview of the process to get these traces is given in Figure 4.3. Delays in this constellation are varying between 70 ms and 100 ms.

We also consider that apart from the forward LMS link, there are no transmission errors. This assumption eases the impact of reliability schemes analysis on the

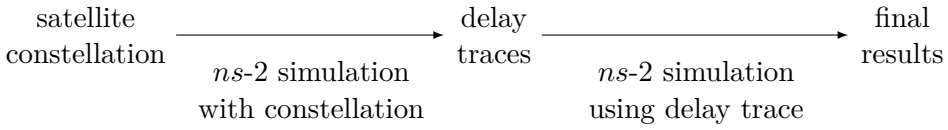


Figure 4.3 – Generating delay traces and playing them in *ns*-2

forward transmission (*i.e.* from the gateway to the terminal) and remains consistent as Inter-Satellite Links (ISL) do not usually exhibit significant transmission errors. Finally, we consider that the routing within the constellation prevents congestion drops inside the constellation. Thus, in our scenario, the bottleneck of the network is the LMS channel.

The size of the queues in the model have been set following Equation (4.1). This formula is a widely used rule-of-thumb, working well with one single flow performing. In case of several flows sharing the buffer, an improvement of this formula is used in Chapter 6.

$$B = \overline{RTT} * C \tag{4.1}$$

To counteract the high error rate, Adaptive-HARQ is used on the LMS channel, in order to optimize the bandwidth utilization. This module did not exist in *ns*-2, and had to be implemented. In the next paragraph we explain the main characteristics of Adaptive-HARQ implementation.

4.2.2 Implementing Adaptive-HARQ in *ns*-2

Network Simulator 2 (*ns*-2) [6] is an open-source discrete event network simulator widely used by the research community. The latest version, *ns*-2.35, still proposes further functionalities than the new simulator, *ns*-3 [67], which is still in development, and is more documented, leading to the choice of this version. Furthermore, *ns*-2 provides a better support for LEO satellite networks. To date, an extension of *ns*-3, *sns*-3 [68], is available allowing to simulate satellite networks. However, this extension only proposes the use of GEO satellite, which is out of the scope of our study.

Other simulators also exist such as *OPNET* (expensive and not open-source) or *OMNeT++*. Despite the good performance of *OMNeT++*, *ns*-2 has been chosen for its high number of functionalities suited to the works needed in this thesis: LEO satellite constellation, varying delays or easy implementation of new modules.

While having the simulation scripts (nodes, protocols, etc.) written in OTCL,

the core of *ns*-2 is written in C++. Several object types are implemented in this simulator representing the main modules of the simulator, for example:

- node, representing a computed host or a router;
- queue, modeling the buffering in the routers;
- trace generator, outputting the data on trace files;
- agent, responsible of creating, forwarding or destroying packets. They can be used for routing or to represent transport layers. They are often linked to a node.

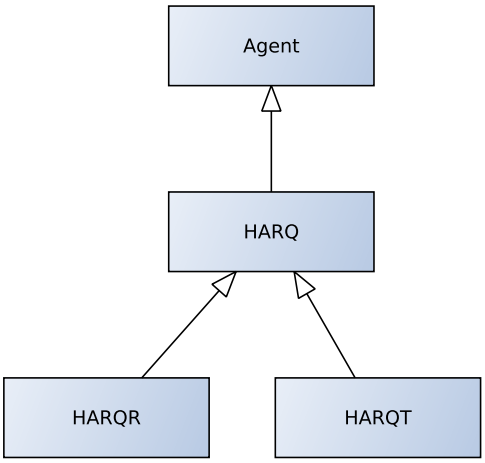


Figure 4.4 – Class diagram of HARQ implementation in *ns*-2

The implementation of **Adaptive-HARQ (A-HARQ)**, detailed in Subsection 3.3.3, in *ns*-2 defines two new agents, $HARQT$ and $HARQR$ representing the sender and receiver sides of A-HARQ, as shown in Figure 4.4. These agents override the *recv*() functions (called when a packet is received) to specify how to manage the packets with A-HARQ:

- the *recv*() function of $HARQR$ receives either a new packet, or redundancy bits corresponding to a packet already stored. In both cases, the function computes and updates the received Mutual Information received for this packet. Then if the packet is decodable, it forwards the decoded packet and sends a positive Ack to $HARQT$, but if it is not decodable, a negative Ack is sent to the sender, and the packet remains stored while it does not reach 3 retransmissions. If this is already the third retransmission, the packet is dropped;

- the *recv*() function of *HARQT* receives the new packets to be sent on the LMS channel, and the positive and negative Acks from *HARQR*. When a new packet is received, this function computes the number of bits to send at each transmission, and sends the first part of the coded packet. Then if a positive Ack is received, the packet is removed from the buffer as it has been decoded. If the Ack is negative, the next redundancy bits are sent if the retransmission limit is not reached, or the packet is discarded if this limit is reached.

Other functions are implemented to send the Acks, to store the packets being decoded or for the improvements explained later in this chapter.

The buffer storing the packets being decoded has been sized to never be full: in the worst case (each packet needs 3 retransmissions to be decoded), 360 packets need to be stored in the buffer in our scenario. The buffer has a size of 500 packets, which is high enough to prevent buffer overflow.

To go on with *ns*-2, we use *ns*−2 *TCP*−*Linux* [69], where the simulator uses the TCP congestion control from the Linux kernel. This allows to have results closer to real Linux implementation, and also a higher simulation speed in some cases where *ns*-2 performs bad performance. Moreover, it allows to use TCP congestion controls algorithms not present in the defaults *ns*-2 implementation such as CUBIC or TCP Hybla.

4.2.3 TCP versions and parameters

We study the two main TCP variants used for terrestrial transmissions: TCP NewReno (denoted NR in the Figures and Tables) and CUBIC. As it is impossible to isolate the space segment, standard TCP protocols need to be used. Even though TCP NewReno is less and less populating the Internet, this protocol is of interest for our problematic: it brings a reference case for our simulations and its basic functionalities make its behavior easier to understand than more sophisticated protocols. We also consider CUBIC since its error recovery is more aggressive than TCP NewReno and it is enabled by default in several Operating Systems: GNU/Linux, OSX systems (since `10.9`) and Windows (since `Windows 10 Fall Creators Update`).

We test TCP/SACK performance with standard GNU/Linux parameters. The first one is Duplicate SACK (D-SACK), which is enabled by default on GNU/Linux systems. D-SACK allows TCP sender to differentiate in some cases, if a Fast Retransmit is due to a reordering or to a packet loss. Thus, the TCP sender can

adjust the duplicate acknowledgment threshold [66], which is usually 3 by default in GNU/Linux.

We also activate Delayed Acknowledgment, as it is mainly activated by default in the real systems. However, we do not make a detailed analysis in this chapter as this point is deeply studied in Chapter 5.

4.2.4 Simulation scenario

Each simulation lasts 600 s with one single TCP flow performing. This duration allows TCP to enter in its established regime, and gives enough time to get the impact of the satellite movements and A-HARQ on the transport layer.

Concerning the LMS link, the quality of this channel varies by setting an reference SNR ranging from 7 dB to 13 dB. During the simulations, the link quality changes over time around this SNR reference value according to a trace file representing the evolution of the LMS channel. For all other links (return link included), we assume that there are no errors to ease the interpretation of the results.

4.3 Studying the impact of out-of-order packets

In this section we investigate TCP performance in our scenario, and understand the impact of out-of-order packets on the protocol. Results of the main TCP metrics are shown in Table 4.1. This table shows the results without the use of Delayed Acknowledgments. We also add that in this section and the next ones, the numbers of RTO, DUPACK and spurious retransmission have been normalized by the number of packets sent, in order to have comparable results between the different graphs and tables.

SNR (dB)	TCP Goodput (kb/s)		A-HARQ Success (%)		RTO (%)		DUPACK (%)		Spurious (%)	
	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC
7	219	256	95.44	95.37	0.74	0.25	2.58	2.34	2.71	2.24
8	276	356	97.55	96.92	0.27	0.05	2.42	2.18	2.67	1.68
9	300	382	97.97	97.49	0.17	0.02	2.41	2.15	2.83	1.68
10	307	458	98.50	98.40	0.09	0	2.60	1.86	3.00	1.39
11	340	473	98.84	98.86	0.05	0	2.25	1.81	2.76	1.51
12	354	486	99.18	99.12	0.02	0	2.18	1.64	2.77	1.54
13	380	521	99.41	99.40	0.01	0	2.08	1.50	2.63	1.32

Table 4.1 – Performance of the main TCP metrics without reordering mechanism

In Table 4.1, the RTO and DUPACK columns represent the proportion of recovery events (due to link errors or congestion events), and not the number of packets retransmitted, as there can be more than one packet retransmitted for each loss detected. We divided the number of times TCP received a request for retransmission by the total number of packets sent. For example, when $SNR = 7\,\text{dB}$, during 600 s, CUBIC transmitted 21406 packets, the RTO timer expired 54 times and 500 retransmissions have been triggered due to DUPACK (DUPACK events). We have also measured 480 spurious transmissions. Thus, the proportion for RTO is 0.25 %, for duplicate acknowledgments is 2.34 % and for spurious is 2.24 %.

We observe that TCP performance in this scenario is very poor, for both NewReno and CUBIC, the goodput being of only a few hundreds of ks/s, whatever the value of reference SNR, whereas we could expect a goodput of 40 Mb/s on a lossless network (measured with another *ns*-2 simulation without losses on the LMS channel and confirmed with Equation (3.3)). Moreover, the high number of out-of-order packets due to the topology of the constellation and the use of reordering mechanisms causes a lot of spurious retransmissions, triggered by DUPACK. Even with good channels, *i.e.* high SNR, the proportion of DUPACK and spurious retransmissions remains high, whereas we could expect a lower value of these metrics when the channel quality increases. These out-of-order packets are interpreted as congestion losses by the TCP sender, ignoring that some packets are just delayed. The packets are then spuriously retransmitted and the congestion window is halved, explaining this bad performance. This leads to a need to minimize the out-of-order packets in order to improve TCP performance.

In the next section, we investigate the impact of adding a reordering mechanism after A-HARQ, to mitigate the impact of out-of-order packets and let TCP exploit the available capacity.

4.4 Mitigating the impact of out-of-order packets

Both delays changing in the constellation and reliability mechanism generate out-of-order packets to the TCP receiver. Concerning the A-HARQ scheme, a packet p needing one more retransmissions is delayed by approximately 20 ms. During this time period, a lot of packets sent later arrive before p.

One simple solution is to add a reordering mechanism after A-HARQ in order to deliver in-order the packets to the transport layer, and improve TCP performance. An overview of the whole system with A-HARQ and the reordering mechanism is given in Figure 4.5

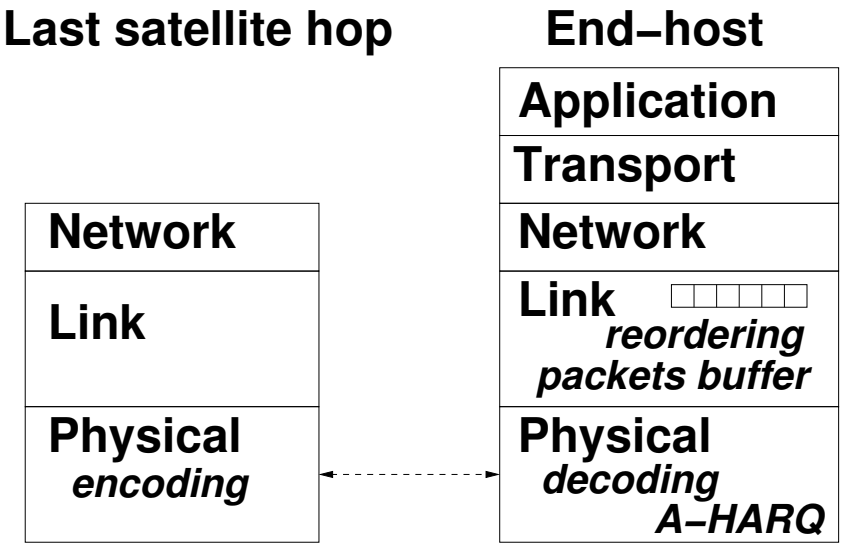


Figure 4.5 – Proposal to add a reordering mechanism after A-HARQ

4.4.1 Adding a reordering mechanism

The reordering mechanism is composed of a buffer storing out-of-order packets. The packets are forwarded from the buffer to the upper layer in-order following their sequence number. There are some issues that may happen and which need to be managed:

- the reordering buffer has a size of 125 packets. This value is computed to find an acceptable size while avoiding TCP timeouts for packets waiting a too long time in the buffer. This optimal value has been computed based on the throughput and retransmission delay on the LMS link. Even if this rarely occurs, the buffer may sometimes be full, implying packets drop. In this case, the solution is to empty the buffer and to forward to the upper layers all packets stored in order to let TCP recover losses as soon as possible;

- when the LMS channel quality is very bad, some packets may be not decoded by A-HARQ, even after 3 retransmissions. In this case the corresponding packet is dropped, and it is impossible to know the sequence number. The reordering mechanism cannot guess the sequence number of the missing packet, and proceed to a correct reordering. Then the solution is also to empty the buffer and send forward all the packets stored. Thus, TCP reacts to the missing packet(s) and triggers a retransmission as soon as possible.

Finally, in both cases the rationale is to speed up TCP recovery procedure, by

sending all the packets to the transport layer, and let TCP recover the missing packets as fast as possible using its loss detection algorithms.

An overview of the reordering mechanism managing a decoded packet is given in Figure 4.6. This case works for one flow, comparing the sequence number of a decoded packet with the sequence number expected by the buffer. When several flows are using the mechanism, there is an expected sequence number for each flow id.

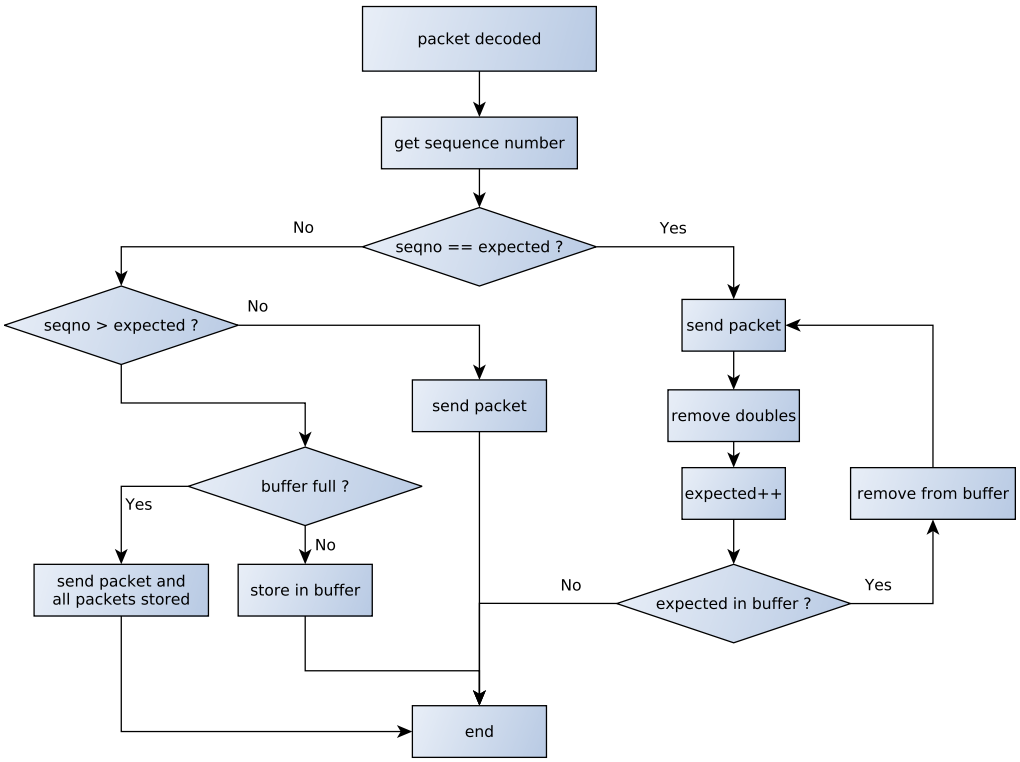


Figure 4.6 – State diagram describing the reordering mechanism

From the TCL script setting the simulation, several parameters of HARQ and reordering mechanism can be set, to consider several cases, for example the activation of the reordering mechanism. Moreover, we set in the simulations the reference SNR in the HARQ module. In real systems, this reference SNR would be estimated by HARQ.

4.4.2 Results with the reordering mechanism

Adding a reordering mechanism greatly improves TCP performance for both TCP NewReno and CUBIC as shown in Table 4.2. Particularly, TCP goodput clearly

SNR (dB)	TCP Goodput (kb/s)		A-HARQ Success (%)		RTO (%)		DUPACK (%)		Spurious (%)	
	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC	NR	CUBIC
7	422	435	95.37	94.95	0.62	0.59	0.52	0.56	1.45	4.00
8	531	631	96.92	97.18	0.41	0.29	0.40	0.50	1.20	4.65
9	609	829	97.49	97.83	0.30	0.22	0.35	0.45	1.27	4.47
10	695	1135	98.40	98.38	0.23	0.16	0.27	0.40	1.21	3.94
11	830	1394	98.86	98.94	0.19	0.09	0.25	0.32	1.13	4.02
12	896	1749	99.12	99.22	0.14	0.06	0.20	0.28	1.20	4.04
13	1051	1955	99.40	99.47	0.10	0.05	0.17	0.25	1.02	3.92

Table 4.2 – Performance of the main TCP metrics with reordering mechanism

increases, as shown in Figure 4.7, and the number of retransmissions decreases. However, we observe a slight increase of the number of timeout events, without affecting the benefits brought by reordering on global TCP performance. This is due to losses previously due to DUPACK (Figure 4.9) are now due to RTO (Figure 4.8), but the total number of loss events finally decreases with the reordering mechanism.

The proportion of spurious retransmissions also increases (Figure 4.10) with the addition of the reordering mechanism, this increase is mainly due to better environment for TCP, making it more opportunistic. With these better network conditions, congestion appears on the network, leading to these spurious retransmissions.

As a result, thanks to the conjointly use of the A-HARQ and reordering mechanism, TCP performance has been highly improved.

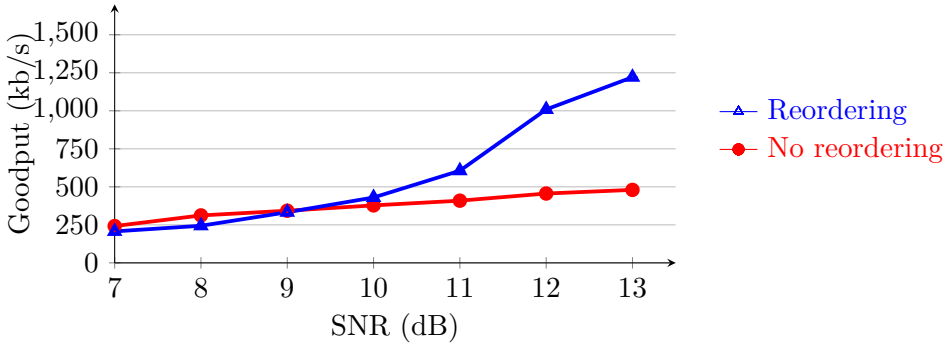


Figure 4.7 – Impact of reordering mechanism on end to end goodput (CUBIC)

We recall that we only presented results with TCP NewReno and CUBIC. However, the same trend has been observed with TCP Westwood or TCP Hybla and should be also observed with other TCP variants. When comparing TCP NewReno and CUBIC, we observe a better performance for the second one. Thus, CUBIC

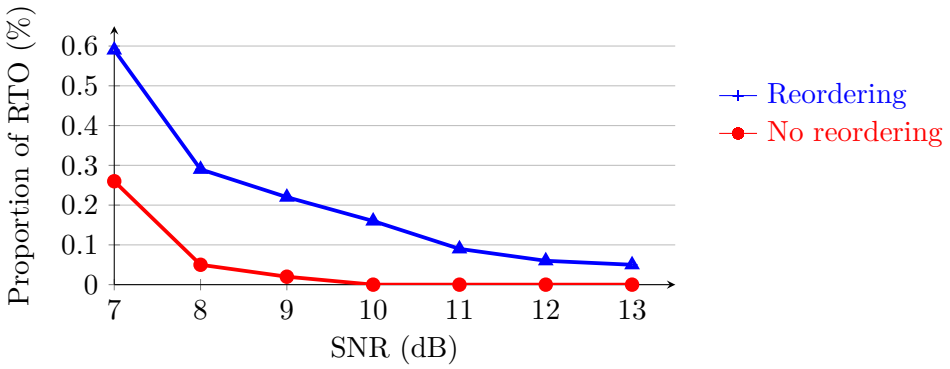


Figure 4.8 – Impact of reordering mechanism on RTO (CUBIC)

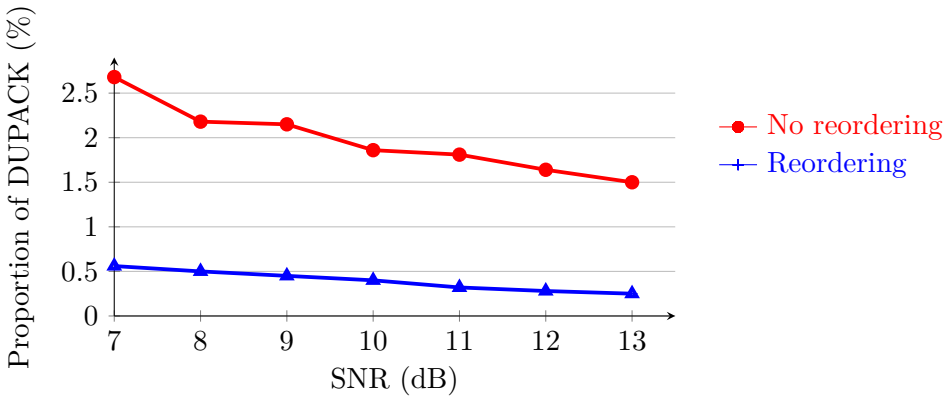


Figure 4.9 – Impact of reordering mechanism on DUPACK (CUBIC)

is well-known to achieve better performance over high-delay bandwidth product networks, explaining the higher performance obtained by this protocol compared to TCP NewReno [8].

4.5 Performance analysis

The results presented in Section 4.4.2 highlight the interest for mitigating the number of out-of-order packets linked to the use of link layer reliability schemes and LEO satellite constellation. The mechanism proposed induces a high decrease of the number of DUPACK, as expected, by TCP. However, the use of reordering mechanism brings additional delay in the network, and increases the number of TCP timeout. This increase does not affect TCP performance for high SNRs, the total number of retransmissions is lower with reordering, increasing TCP goodput. This performance benefit is not only a gain for the use of the expensive satellite capacity, but also for the end-to-end latency. At last but not least, this makes it

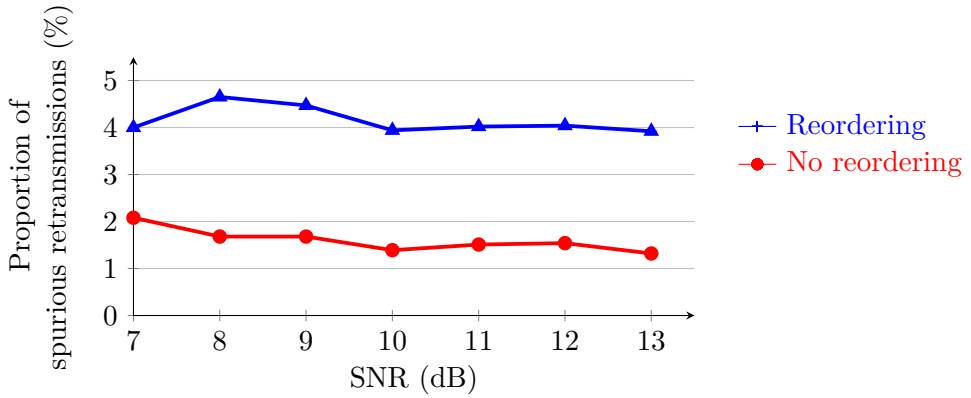


Figure 4.10 – Impact of reordering mechanism on spurious retransmissions (CUBIC)

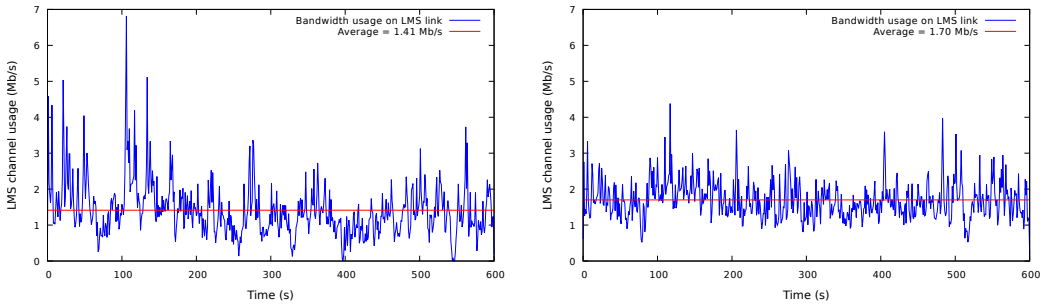
easier for delay tolerant services to respect their delivery rate constraints. However, we can see in Figure 4.7 that with low SNRs, the higher delay totally mitigates the improvements brought by the decrease of the number of DUPACKs.

Once we have seen the impact of reordering mechanism on TCP performance, we need to evaluate the impact of other TCP parameters such as Duplicate SACK (D-SACK). We evaluate its impact on TCP performance, and did not see significant goodput improvement with or without this mechanism, we only observe a gain of 3% in the best case, but it can also slightly decrease in some other cases. It is enabled by default on GNU/Linux, and can stay enabled regarding the performance obtained, there is no need to change the values of this TCP parameter. The other TCP metrics which are DUPACK, RTO, and spurious retransmissions are not significantly impacted by D-SACK.

Moreover, when the reordering mechanism is activated and the channel conditions are good enough, we observe congestion appearing on the network, and packet drops due to this congestion and not to satellite impairments. This means that A-HARQ conjointly to the reordering mechanism recovers enough packets to allow standard TCP protocols to work in good conditions.

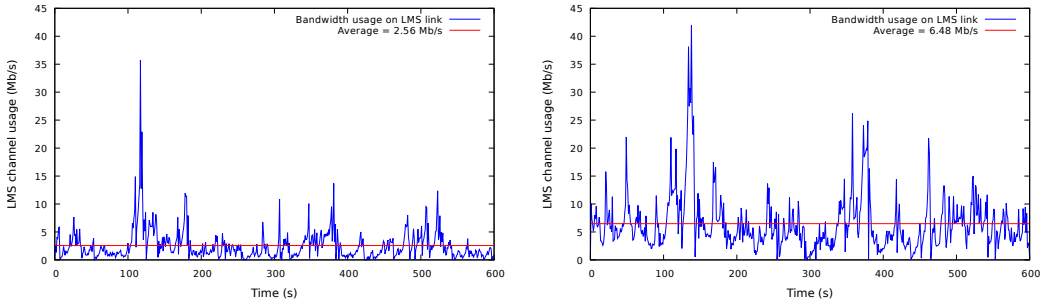
Finally, the LMS channel usage is improved with the reordering mechanism, especially for high SNRs. We can see in Figures 4.11 and 4.12 that the reordering mechanism improves the LMS channel usage, and thus the number of packets that are transmitted. We also note that the maximum LMS link capacity can be reached with the reordering mechanism, with an average value for $SNR = 13$ of 6.48 Mb/s. As a reminder, we set the capacity of this link at 50 Mb/s. This usage corresponds to the real number of bits flowing through this link, including the redundancy bits and the retransmissions by A-HARQ. Obviously, the channel is not totally used,

allowing other flows to share this link. We will see in Chapter 6 that further improvements can be made when a lot of flows are sharing the LMS link.



SNR=8 dB SNR=13 dB

Figure 4.11 – Usage of the LMS link, using CUBIC, without reordering



SNR=8 dB SNR=13 dB

Figure 4.12 – Usage of the LMS link, using CUBIC, with reordering

Finally, we showed the results we obtained for Adaptive HARQ tends to counteract the effects of out-of-order packets caused by this reliability scheme. However, all HARQ schemes imply varying delays and out-of-order packets. Thus, our conclusion is still correct with any HARQ scheme. Thus, adding a reordering mechanism after HARQ should be the standard for all HARQ mechanisms, in order to always guarantee good TCP performance.

However, Figure 4.13 highlighted the significant impact of a TCP parameter, activated by default, called DelAck. When the reordering mechanism is enabled, using DelAck highly decreases TCP performance, compared to the case with reordering but no DelAck. However, when no reordering is present, DelAck does not significantly change TCP performance, with a small goodput increase for NewReno and no change for CUBIC. This result is deeply studied in Chapter 5. On another hand, if we consider DelAck always activated, the reordering mechanism still increases TCP performance, but in lower magnitude than without DelAck. Thus, for

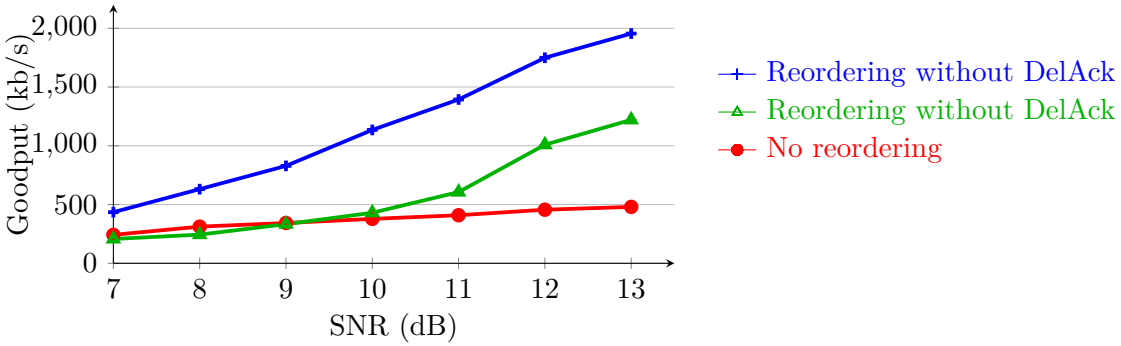


Figure 4.13 – Impact of reordering mechanism on end to end goodput (CUBIC)

transmissions where DelAck activation cannot be negotiated, the reordering is still useful for TCP performance.

Recent works also highlighted the need to take into account the impact of out-of-order packets in Mobile Networks [70]. In this context, as for satellite context, out-of-order packets significantly decrease TCP performance, leading to the need to adapt TCP to this problem.

4.6 Conclusion

The impact of out-of-order packets has been studied in this Chapter. It showed the need to introduce a reordering mechanism conjointly to HARQ to mitigate the effect of out-of-order packets caused by this reliability scheme and the satellite constellation. With this solution, TCP performance is highly improved.

However, even if some TCP options such as D-SACK do not have any impact on TCP performance, we highlighted a significant negative impact of DelAck on TCP goodput. This result needs to be fully understood. The next Chapter is dedicated to study this TCP option in our context of satellite transmissions.

Chapter 5

Impact of DelAck on TCP performance

Contents

5.1 On the need to study the impact of DelAck

Delayed Acknowledgment (DelAck) [10] has been introduced to improve TCP performance, but has only been deeply studied on wired networks. DelAck reduces the number of acknowledgments and thus, on these kinds of networks, the feedback path load. This lower number of Acks also reduces the CPU load [11]. These benefits have led RFC1122 [10] to recommend the activation of this option, and several systems such as Windows OS, MacOS or GNU-Linux OS enable by default DelAck.

However, in the context of LEO satellite transmissions, where the error rate is high and the transmission delay greatly varies because of the topology and the reliability schemes, the impact of DelAck needs to be deeply studied. We show in this Chapter that the conjoint use of HARQ with DelAck can be counterproductive, confirming the concerns we raised in Chapter 4. Actually, only a few studies analyzed the impact of DelAck in such environment. J. Chen *et al.* [71], studied the impact of DelAck on TCP over wireless links and showed that their impact on TCP performance depends on the topology used and the path length. They conclude that activating DelAck does not always improve TCP throughput. L. Wood *et al.* [72] has also raised some concerns in the context of satellite systems, where the losses are not only due to congestion but also to transmission errors. These reasons have led to the need to investigate the impact of DelAck in the context of LEO satellite constellations, to determine if the default activation of DelAck is relevant in this context.

5.1.1 DelAck algorithm and TCP

As explained in Chapter 3, Delayed Acknowledgment allows to combine two packets to generate only one acknowledgment informing the good reception of both packets. There are two requirements to generate a DelAck:

- R1: the packets must be in-order. When a out-of-order packet is received, TCP sends instantly an Ack to inform the sender as quickly as possible a potential loss in the network;
- R2: the two in-order packets must arrive within a fixed time window. This window is often 40 ms (but can be 200 ms with Windows OS) which is a trade-off letting enough time to TCP to wait for the second packet while avoiding TCP timeout (RTO). Thus, in our context, DelAck adds more delay to a path where the transmission delay may be already important due to the use of reliability mechanisms and satellite constellation.

These requirements show that the satellite network characteristics seem not optimal for the use of DelAck, confirming the need to study its impact on TCP when transmitting on a satellite constellation.

Concerning the TCP variants used, we investigate the impact of DelAck on TCP NewReno and CUBIC, and also on variants developed for space transmissions such as TCP Hybla. As explained in the previous chapters, TCP NewReno and CUBIC are the two main variants deployed over the Internet, and our topology allows the use of such variants, due to the RTT ranges. However, we also chose to study TCP Hybla because it has been designed for space transmissions, and because its congestion control algorithms have a different evolution compared to TCP NewReno or CUBIC.

5.1.2 Simulation scenario

We keep the same topology than in Chapter 4, emulating a LEO satellite constellation composed of 66 satellites at an altitude of 800 km, using a 3 nodes model and varying delays. On the LMS link, we still have the Adaptive-HARQ module to deal with the channel impairments. We study the impact of DelAck in two cases, depending on the activation or not of the reordering mechanism.

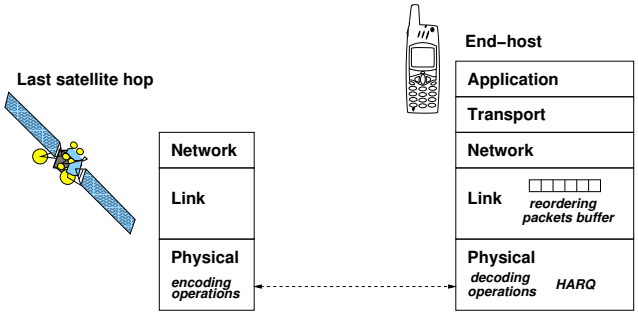


Figure 5.1 – Description of the low layer mechanisms

The channel quality varies around a reference SNR, ranging from 7 dB to 13 dB. During the simulations, the link quality changes over time around this SNR reference value. Each simulation lasts 600 s with one single TCP flow performing. We kept the 40 ms DelAck timer value in the *ns*-2 simulations. This value is often the default value on GNU/Linux systems, and is a compromise between a waiting time high enough to receive the second packet but not too important to avoid TCP timeout.

5.2 Study of the impact of DelAck without reordering

Firstly, we analyze the impact of DelAck without the use of the reordering mechanism.

5.2.1 Impact of DelAck on TCP performance

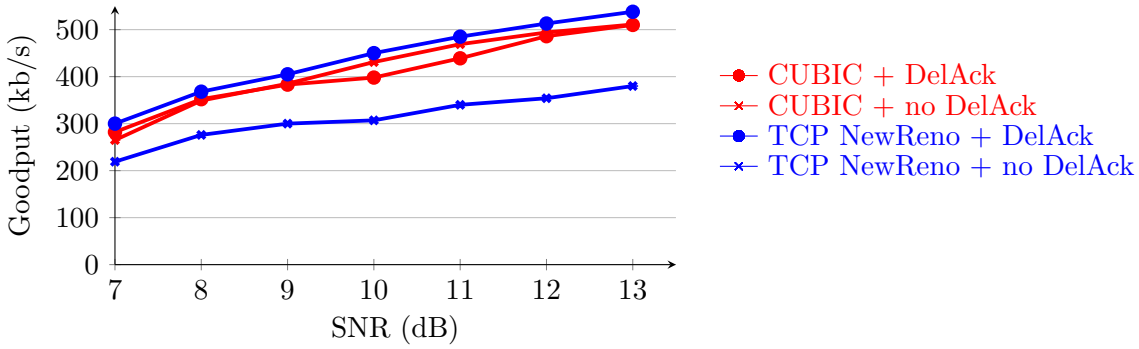


Figure 5.2 – Impact of DelAck on end to end goodput

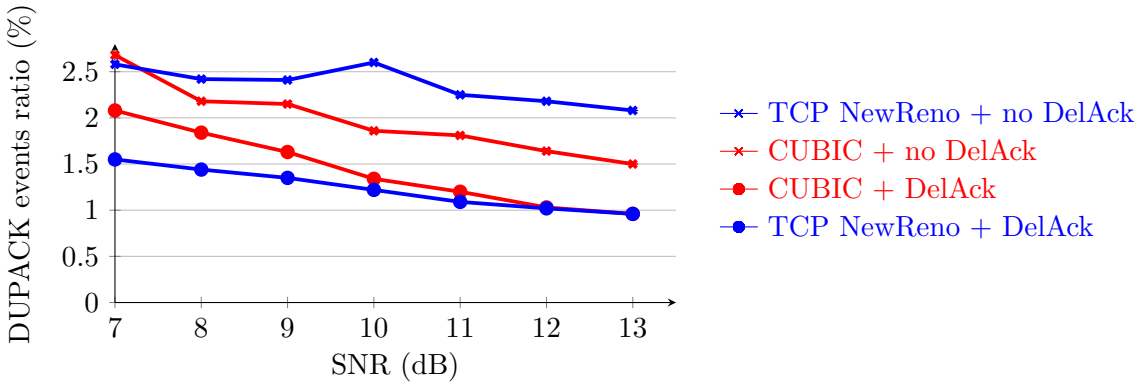


Figure 5.3 – Impact of DelAck on DUPACK events

As shown in Figure 5.2, DelAck significantly improves TCP performance with NewReno, with a goodput gain up to 40 %, unlike CUBIC where the goodput is approximately the same. This scenario has also been driven with TCP Hybla. In this case, we observe a strong decrease of TCP goodput, up to 40 %, as shown in Figure 5.6.

This trend can also be seen on the other TCP metrics such as DUPACK events (Figure 5.3), RTO events (Figure 5.4) and spurious retransmissions (Figure 5.5). With TCP NewReno, DelAck improves all these metrics, explaining the good performance, but with CUBIC, there is either no change or only a small improvement

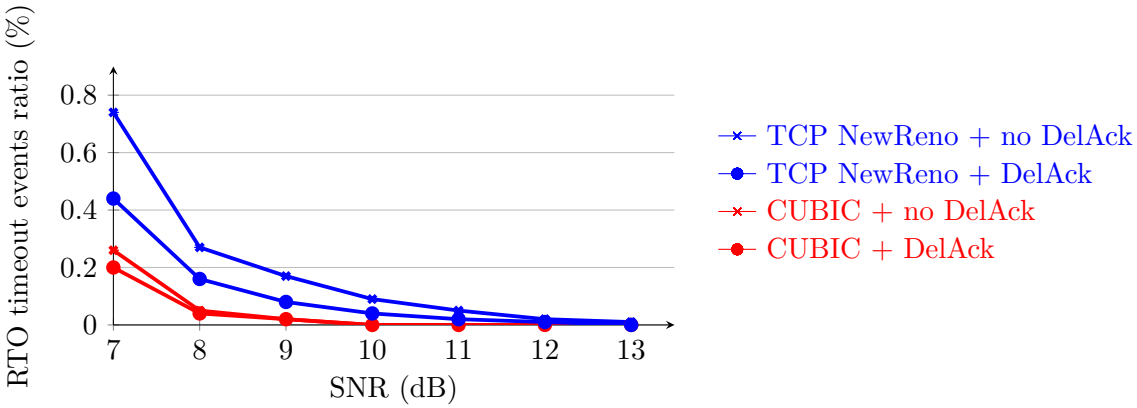


Figure 5.4 – Impact of DelAck on RTO events

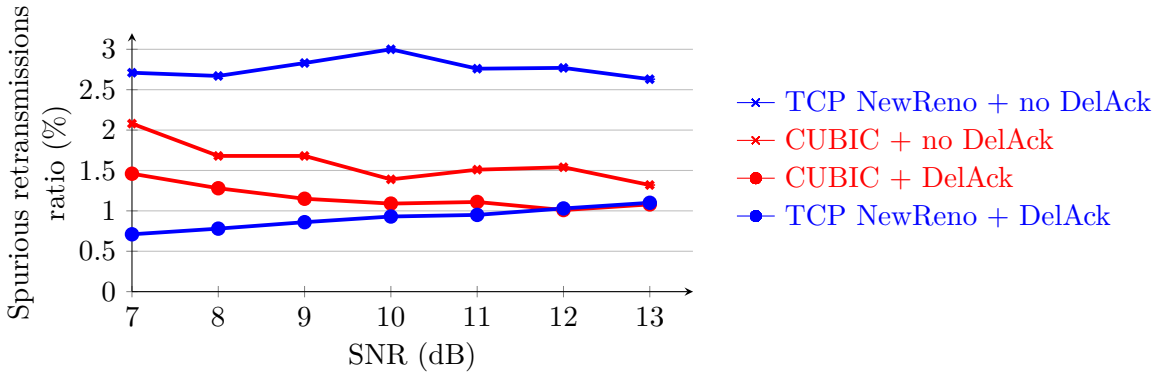


Figure 5.5 – Impact of DelAck on spurious retransmissions

of these metrics.

Finally, we have three different behaviors depending on the TCP variant used: an improvement with TCP NewReno, a strong performance decrease with TCP Hybla, and no significant change with CUBIC.

5.2.2 Analysis

To understand these different TCP behaviors when DelAck is activated, we need to study the impact of DelAck on the congestion window growth. The window is generally updated when an acknowledgment is received, and the use of DelAck, increasing the waiting time between two acknowledgments, slows down this growth due to the lower number of acknowledgments [72].

This evolution can be seen in Figure 5.7, where each curve has been independently generated using one single flow between two nodes, without any supplementary error loss introduced nor HARQ. This simple case allows to highlight the

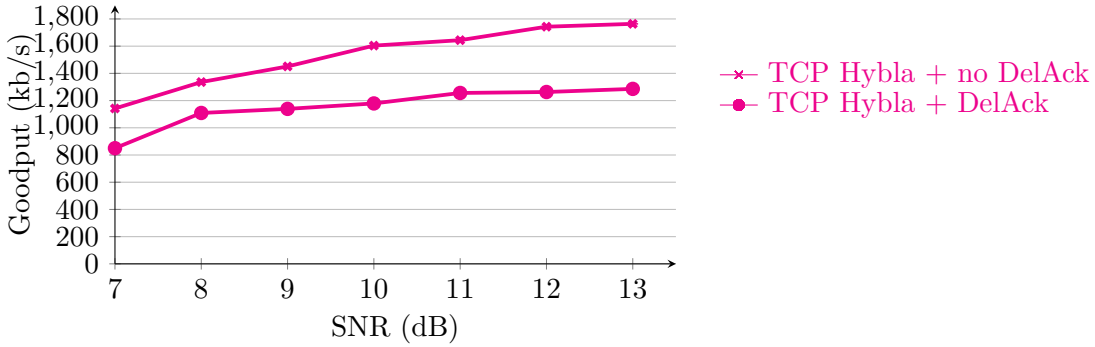
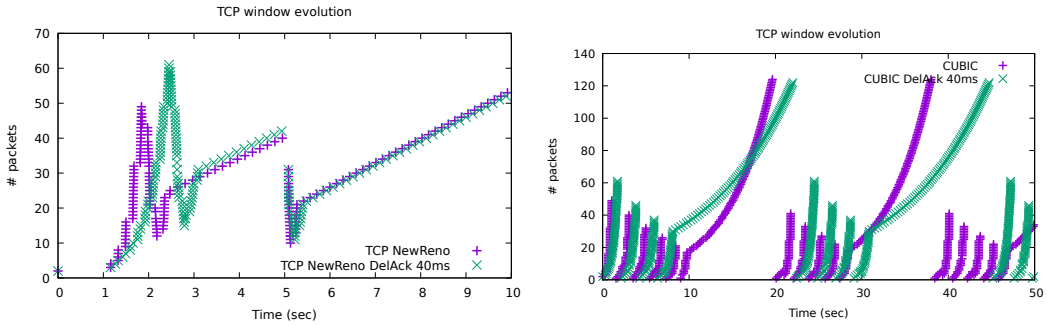


Figure 5.6 – Impact of DelAck TCP Hybla on end to end goodput

impact of DelAck on the congestion window growth. This growth is always lower in Slow Start phase whatever the variant used.



With TCP NewReno With CUBIC

Figure 5.7 – Evolution of TCP congestion window

As presented in [47], such pacing can improve TCP performance on congested networks by delaying the filling of the buffer at the bottleneck and allowing to find a larger optimal congestion window. However in the context of LEO satellite constellations, losses are due to transmission errors and not to congestion, and then pacing only delays transfer and loss detection signals [50], resulting in a decrease of the goodput.

We also observe that depending on the TCP variant used, DelAck sometimes slows down the growth in Congestion Avoidance phase. TCP NewReno is not impacted by DelAck in CA, because the evolution of its *cwnd* is linear and based on the number of packets acknowledged. On another hand, CUBIC congestion window evolution is based on a temporal function, in which the parameters are set depending on the state of the transmission during a loss event. Particularly, the

evolution of the congestion window in CA is as follows:

$$cwnd = W_{max} + C(t - K)^3 \tag{5.1}$$

In this equation, K is proportional to W_{max}, and knowing that W_{max} is higher with DelAck due to the pacing induced, the growth of *cwnd* with DelAck is also lowered with DelAck.

Moreover, transmission errors often occur on LEO satellite constellations, sometimes not recovered by reliability schemes, triggering more retransmission. The high jitter due to the topology and the use of HARQ also causes a high number of spurious retransmissions when no reordering mechanism is present. The high number of loss events results on TCP oscillations between Slow Start and Congestion Avoidance phases, where the congestion window is growing slower [49], counteracting the improvements that may have been brought by DelAck.

This higher number of retransmissions can be observed in Table 5.1. We observe that CUBIC triggers more retransmissions than TCP NewReno, implying that CUBIC is more often in Slow Start and Fast Recovery phases. Thus, the low number of retransmissions with TCP NewReno allows this variant to still take advantage of the performance improvement brought by DelAck, whereas with CUBIC, the high number of retransmissions lowers the congestion window growth, decreasing overall TCP performance.

SNR (dB)	Number of retransmissions		
	TCP NewReno	CUBIC	TCP Hybla
7	1068	1333	6969
8	932	1100	16914
9	925	1105	23073
10	789	949	16782
11	748	923	22399
12	742	780	23852
13	706	789	20688

Table 5.1 – Number of packets retransmitted during all the simulation, when DelAck is activated

This trend is also observed with TCP Hybla: the number of retransmissions is higher than TCP NewReno or CUBIC due to the larger congestion window [12]. This high number of retransmissions is a consequence of TCP Hybla congestion

control algorithms, which are able to react correctly to losses, when DelAck is disabled. However, as for CUBIC, the activation of DelAck only lowers the congestion window growth, decreasing this TCP variant performance.

Thus, in our scenario, when a transport protocol is aggressive (*e.g.* TCP Hybla or CUBIC), enabling DelAck results in slower progression of the congestion window and a reduced goodput, whereas enabling DelAck for transport protocols with lower retransmission levels (*e.g.* TCP NewReno) improves the goodput of the connection, by taking advantage of DelAck: reduced number of Acks, lower CPU load or pacing in the buffers.

5.2.3 Optimizing DelAck timer value

To optimize DelAck efficiency, we can find the best DelAck timer value. Actually, the default value is 40 ms, but we can ask if this is still the optimal value in the case of LEO satellite transmissions. Thus, the additional transmission delay brought by the topology and reliability mechanisms can increase the TCP retransmission timeout probability, and adding a too large DelAck timer value may be counterproductive.

However, as shown in Figure 5.8, we observe that 40 ms is still a fair compromise between a sufficient waiting time and the probability of TCP timeout. This Figure has been generated by computing the number of packets received by TCP depending on two parameters: the DelAck timeout value in milliseconds and the reference SNR. Whatever the value of SNR, we observe a maximum of the curve between 30 ms and 40 ms, confirming the value of 40 ms chosen in our simulations.

However, we need to highlight that this value is linked to the topology studied, a LEO satellite constellation, to the capacity of the links and the delays. Other topologies might lead to other optimal values. Other systems are using different DelAck timeout values to have the best performance with their topology.

5.3 Study of the impact of DelAck with reordering

5.3.1 Impact of DelAck on TCP performance

When the reordering mechanism is enabled after HARQ, we observe, as shown in Figures 5.9 and 5.10, that activating DelAck decreases TCP goodput. This is a slight decrease when using TCP NewReno, but a strong one with CUBIC, especially for high SNR values.

There are to interpretations of these results:

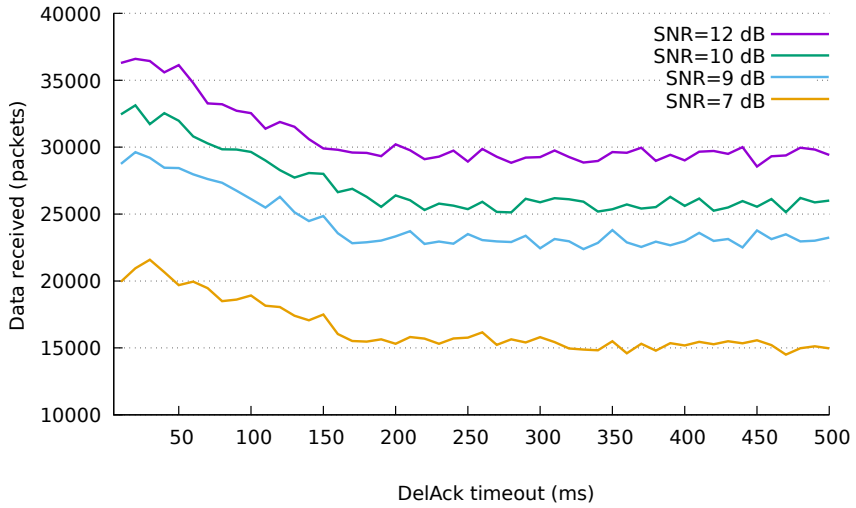


Figure 5.8 – Number of packets received depending on the DelAck timeout value and SNR

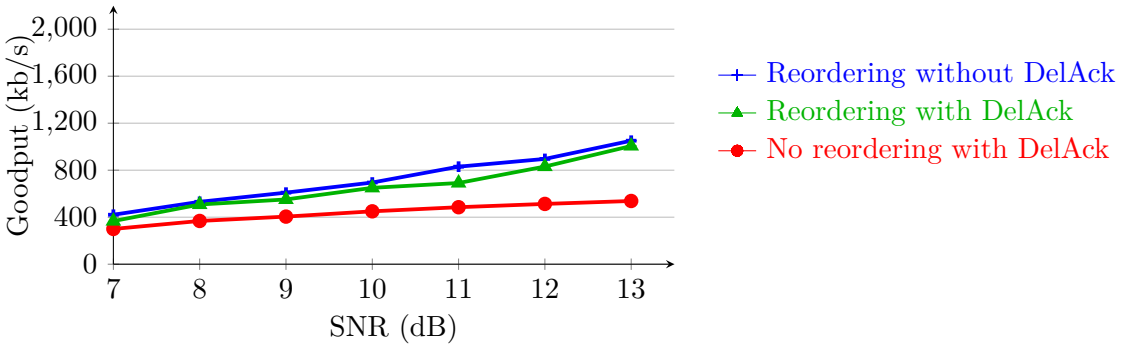


Figure 5.9 – Impact of DelAck on TCP performance with a link layer reordering mechanism with TCP NewReno

- considering the reordering mechanism is always activated, enabling DelAck decreases TCP performance, for both TCP NewReno and CUBIC;

- considering DelAck is always activated, adding the reordering mechanism improves TCP performance, except for CUBIC and low SNR values.

The optimal solution is then to use the reordering mechanism without DelAck. However, there is a need to understand these results, which are totally different than what we observe without reordering. In this case, all TCP variant studied showed a performance decrease with the use of DelAck.

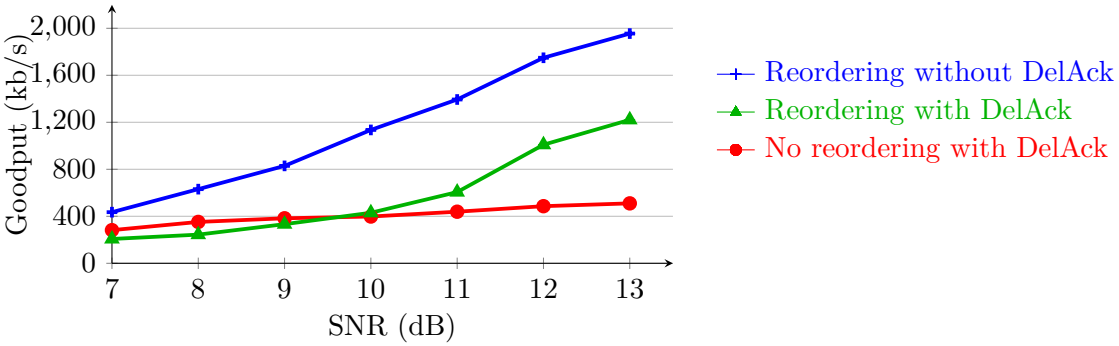


Figure 5.10 – Impact of DelAck on TCP performance with a link layer reordering mechanism with CUBIC

5.3.2 Analysis

To understand the impact of DelAck when the reordering mechanism is enabled, we need to pay attention to the packet transfer delay. We previously saw that the transfer delay may be very important due to the addition of several causes: HARQ, in which the decoding delay is higher when the channel quality is low, the reordering mechanism, and the topology of the constellation. DelAck adds additional delay, by delaying the acknowledgment of the packets.

We measured that the waiting time between two in-order packets forming a DelAck is higher than 10 ms in 10 % of the cases, as shown in Figure 5.11. Thus, the acknowledgment of the first packet of these DelAcks is delayed by the same value, increasing its overall transmission time and the TCP timeout probability. This additional delay brought by DelAck is often proportional to the delay needed by HARQ to get one additional retransmission.

On another hand, the RTO timer value decreases when DelAck is enabled, for both TCP NewReno and CUBIC, giving less time to the packets to reach their destination, also increasing the TCP timeout probability. We recall that TCP RTO timer follows this equation:

$$RTO = SRTT + 4 * RTTVAR \tag{5.2}$$

This algorithm converges towards the RTT, with a small margin to handle the jitter. The more important is the jitter, the higher is the RTO timer to avoid unnecessary TCP timeouts. When this jitter is low, the RTO timer can be close to the RTT to detect a packet loss as soon as possible. In our simulations, we observed that the RTT variation is lower with DelAck. DelAck, combined with the reordering mechanism, smooths the RTT, with an average higher value, but with

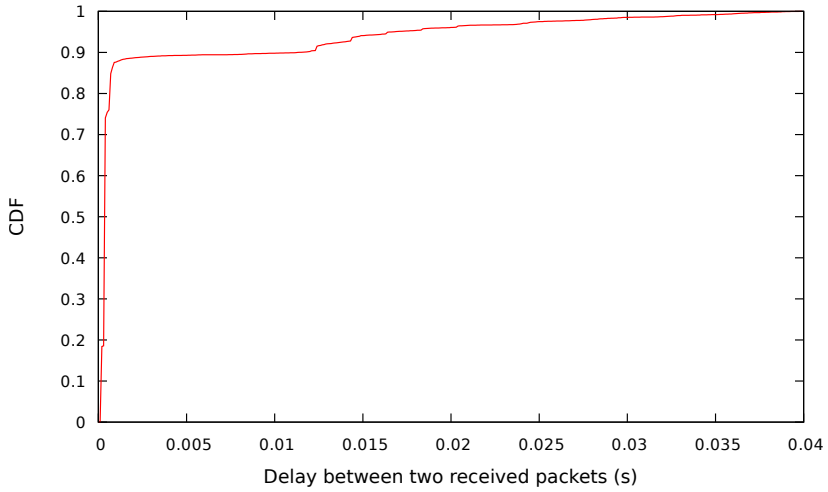


Figure 5.11 – Delay between the reception of two in-order packets forming a DelAck

lower variations. Finally the lower value of $RTTVAR$ takes the advantage over a larger $SRTT$ (mainly because $RTTVAR$ weight in Equation (5.2) is 4 times more important than $SRTT$) and, finally, the RTO timer is lower.

The impact of DelAck on these metrics is confirmed in our simulations, as shown in Figure 5.12, showing a lower average RTO timer value over the whole simulation with DelAck, and then less time for the packets to reach their destination. This timer decrease can be very important with drops up to 25 % with low SNR. Even if this result of timeout value is always lower than 1 s, which is the minimum value proposed in RFC 6298 [45], many systems use lower minimal value, 200 ms for example in the GNU/Linux kernel. Thus, our measured values are consistent with real values.

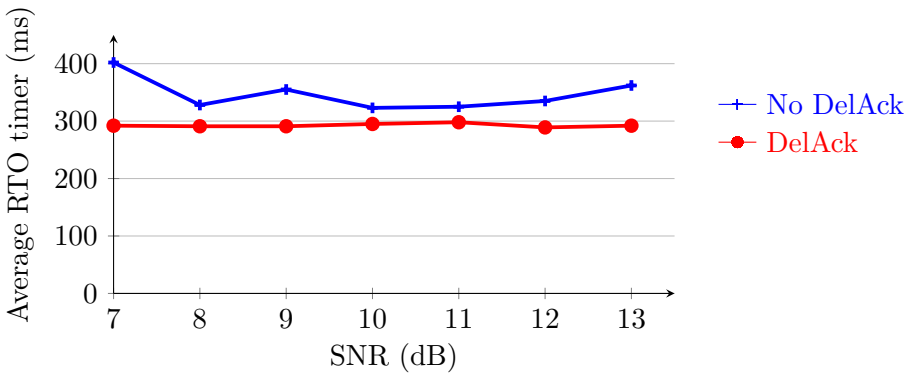


Figure 5.12 – Average RTO timer value using CUBIC with and without DelAck

This lower timer and the larger delay in our scenario has as consequence a

higher number of loss detection due to RTO, as shown in Figure 5.13. Lot of these retransmissions are spurious retransmissions that could have been avoided, uselessly using network capacity. Finally, this causes TCP to trigger more retransmissions and to decrease its congestion window, leading to a performance decrease. The solution is then, when the reordering mechanism is present, and if possible, to disable DelAck, whatever the TCP variant used.

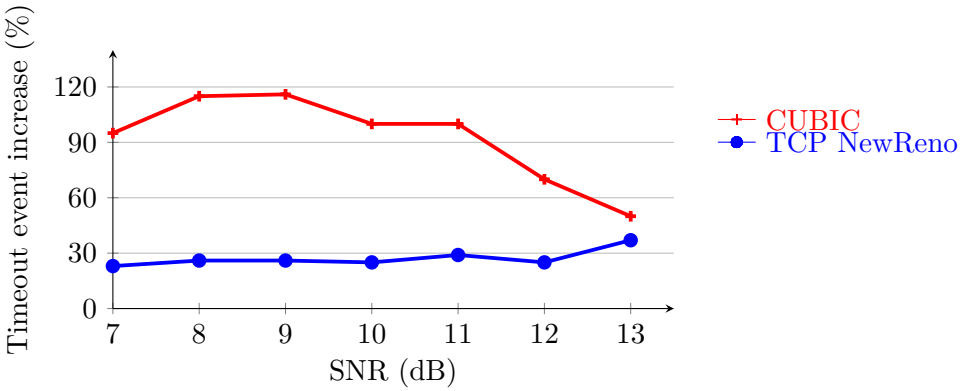


Figure 5.13 – Impact of the activation of DelAck on the retransmission ratio due to timeout when the reordering mechanism is present

5.4 Conclusion

We showed that adding DelAck on TCP transmissions over LEO satellite constellations can have different impact on its performance, depending on the variant of TCP used. When the reordering mechanism is disabled, we recommend to activate DelAck when using TCP NewReno, unlike CUBIC where it can decrease TCP performance in some cases. With this variant, DelAck activation is not mandatory. More generally, TCP variants which are more aggressive and retransmit often such as TCP Hybla are negatively impacted by DelAck, which should be disabled to improve performance.

The impact of DelAck is also always negative when a reordering mechanism is used conjointly with HARQ. In this case the additional delay brought by DelAck and its trend to reduce the RTO timer value highly increases the number of TCP timeouts, decreasing overall TCP performance.

Finally, we also drove simulation with several users transmitting, to assess if the impact of DelAck is still the same on TCP performance. We noted the same tendencies now than with only one user. Thus, all the conclusions brought here can

be extended to the case of several users transmitting in parallel.

More generally, this study highlighted that TCP options that can improve TCP performance on wired networks can be counterproductive in other environments such as LEO satellite networks.

Chapter 6

Scheduling users to improve performance

Contents

6.1 On the need to optimize the LMS channel

The LMS channel is the bottleneck of a LEO satellite constellation network: the high error rate on this link makes the use of reliability schemes necessary, and redundancy must be added to the useful data, decreasing the goodput. In this Chapter, we show that the LMS channel capacity can be optimized, in order to maximize the spectral efficiency of this link, and the user satisfaction.

6.1.1 Actual maximum use of the channel

The HARQ mechanisms allow to correctly receive almost all the packets at a cost of redundancy bits and retransmissions. Adaptive HARQ is designed to select the number of redundancy bits to send depending on the reference SNR, in order to always have the same proportions of packets decoded depending on the number of retransmissions:

- proportion of packets decoded with 0 retransmission $\approx 50\%$;
- proportion of packets decoded with 1 retransmission $\approx 20\%$;
- proportion of packets decoded with 2 retransmission $\approx 15\%$;
- proportion of packets decoded with 3 retransmission $\approx 15\%$.

When several users are transmitting from a mutual last satellite, as shown in Figure 6.1, if no additional mechanism is introduced, each user obtains the above decoding proportions with HARQ. Thus, an important proportion of the LMS channel is used to transmit redundancy and not useful bits.

All the users, sharing the same last satellite, have different mobile receivers on ground, and then different LMS channels. Each LMS channel evolution is different and independent from the others: one user can have momentarily bad channel conditions while another has very good conditions. Thus, an idea is to favor the users when their channel condition becomes good, to optimize the use of the LMS channel capacity: more useful bits sent of the channel and less redundancy bits.

6.1.2 Scheduling the packets to optimize HARQ performance

Such scheduling mechanism aims to send the packets only to the user having momentarily the best channel capacity, as shown in Figure 6.2. The capacity needed to decode each packet is then lower, and more useful data can transmit the link in the same time.

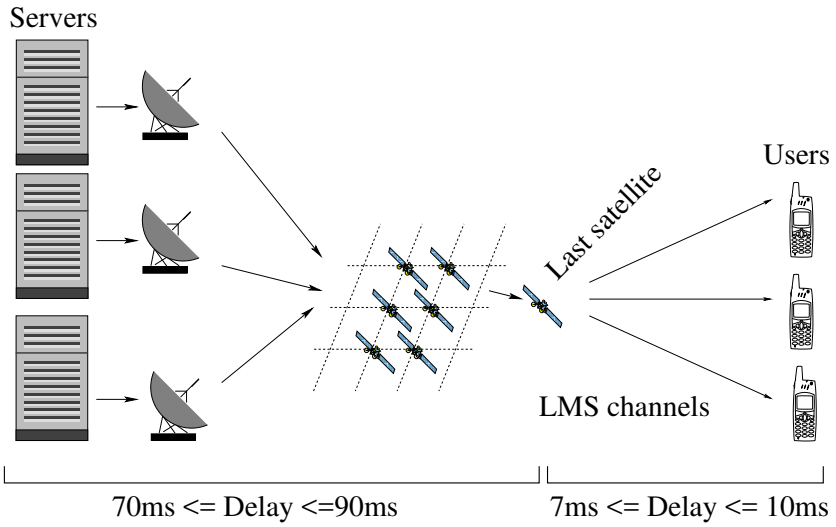


Figure 6.1 – Model for a satellite constellation with several users

Following Figure 6.2, if we consider n users and $C_i(t)$ the capacity of the channel of the user i at time t, then without any scheduling mechanism, the global capacity of the LMS link follows (6.1):

$$C(t) \approx \frac{\sum_{i \in [1..n]} C_i(t)}{n} \tag{6.1}$$

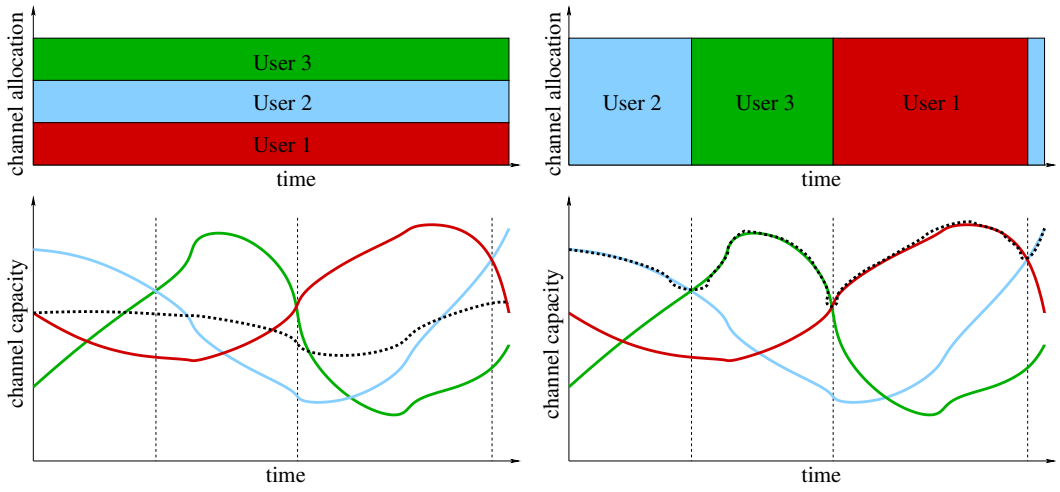
However, with a scheduling policy, where the packets are sent to the user having the best channel, the capacity is now following (6.2):

$$C(t) \approx \max_{i \in [1..n]} C_i(t) \tag{6.2}$$

This second value is clearly higher than the first one, leading to a channel optimization and more useful data to be transmitted.

This proposition induces some questions that need to be considered:

- Q1: how to estimate the channel capacity?
- Q2: how much time a flow having the best channel conditions remains with these good conditions?
- Q3: what is the impact of this low-layer scheme on the upper layers, particularly TCP?
- Q4: what is the expected gain?
- Q5: how to ensure fairness between the users?



Before scheduling After scheduling

Figure 6.2 – LMS channel optimization

Concerning Q1 and Q2, the estimation of the LMS channel capacity in real systems is not instantaneous, but is lower than the transmission delay on the LMS link, *i.e.* 10 ms. During this time period, the channels should not greatly evolve and users having good conditions should still have these good conditions after this delay.

We consider in this study that the channel capacity can be known instantaneously. This is a good assumption considering the evolution of the channel capacity. Moreover, we drove studies with a small delay introduced in the channel capacity estimation, to ensure that there is not significant disadvantage on the performance of the transmissions.

Concerning the expected gain (question Q4), we drove a first Matlab simulation, with three different scenarios: only one user transmitting, several users transmitting without scheduling policy (*i.e.* the packet sent is taken from one random flow), several users transmitting with a scheduling policy (*i.e.* the packet sent is from the flow having the best channel capacity). In all scenarios, we consider the sending rates always high enough to have a packet to send on the LMS channel for each flow. Results with these three scenarios are shown in Figure 6.3.

We can observe a better use of the LMS channel with the use of this scheduling mechanism. This confirms the interest of using such mechanism and hints us on the maximum gain that can be achieved.

In the next sections, we present the different schedulers available and suited for

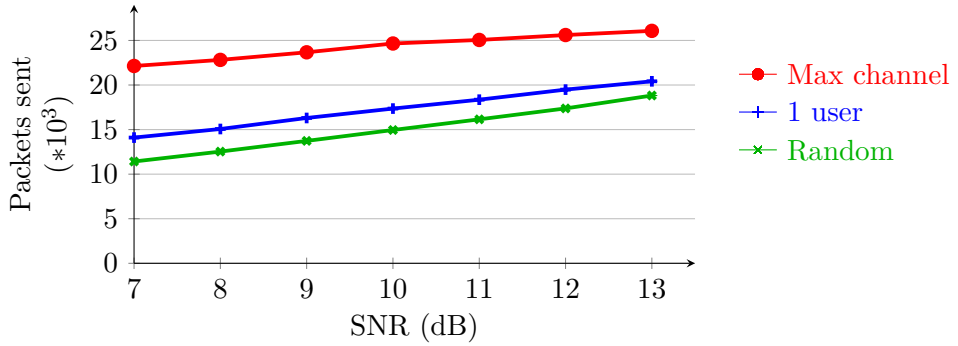


Figure 6.3 – Number of packets sent in the lower layers with different scenarios

our scenario, then we analyze the performance of the selected ones with VoIP and TCP traffics.

6.2 Existing schedulers

Several schedulers have been proposed to optimize the channel capacity and the transmission delay in several contexts. Some studies have already been done to compare these scheduling policies [73, 74]. They can use different metrics to schedule the packets transmission such as the throughput of the channel or the waiting time in the queues. The main algorithms are summarized in Table 6.1.

Algorithm	Metric	Traffic	Complexity
PF [13]	Throughput	best-effort	Very low
M-LWDF [14]	Throughput and time in buffer	Real-time	Low
EXP-PF [15]	Throughput and time in buffer	Real-time and BE	Low
UBMT [16]	Throughput and time in buffer	All	Medium
BBS and BPS [17]	Throughput or time in buffer	Depends on metric	High

Table 6.1 – Comparison of different schedulers

These different schedulers can be grouped into two main categories:

Explicit methods: this category of schedulers uses an evaluation function, depending on one or several metrics, to compute which flow(s) can send packets. The process is as follows:

1. select one or several metrics x_i which are used to schedule the flows;
2. define one evaluation function $f(x_i)$ assessing the state of each flow;

3. select the flow depending on the evaluation functions. For example, the one with the maximum value of this function.

This kind of schedulers are easy to implement and do not require a high computation capacity, which is an advantage in satellite environments, but can be less efficient than other approaches such as Utility Functions. However, some algorithms are still suited for our context and can show good performance.

The most known schedulers using explicit methods are Proportional Fairness (PF), Maximum Largest Weighted Delay First (M-LWDF), Exponential PF (EXP-PF) or Urgency Based Maximum Throughput (UBMT).

Utility Functions: this category needs more complexity and computation capacity, and is linked to Game Theory to schedule the flows. Depending on the metrics selected and a utility function giving the importance of each parameter, this method allows to find a global optimum and how to reach this optimum.

This kind of method allows to have a more flexible solution, and to reach the optimum, but is very more complex to set. Moreover, the computing capacity of the satellites has to be high enough to handle the computation of these algorithms.

The most known schedulers using utility functions are BBFrame by BBFrame Scheduling (BBS) or BBS Periodic Scheduling (BPS).

One of the simplest algorithm in terms of computation complexity is **Proportional Fairness (PF)**, which uses the throughput of the link to send the packet from the flow having the best channel conditions, while minimizing the channel usage. This low complexity and the metric used make PF a good candidate for the optimization of the link. Packets entering the scheduler are stored in different queues based on the user's destination, as shown in Figure 6.4.

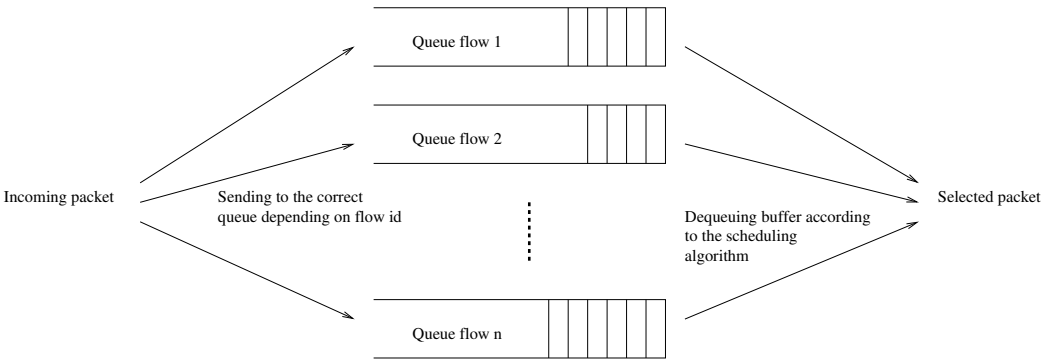


Figure 6.4 – Scheduling the packets and storing in different buffers

There are as many queues as there are destinations in the scheduler. The scheduling of the packets with PF is made as follows: every time a new packet can be transmitted, the scheduler computes the following value (Equation 6.3) for all the queues i:

$$f_i(r) = \frac{r_i(t)}{\overline{r_i}(t)} \tag{6.3}$$

In this equation, $r_i(t)$ represents the throughput of the channel i at time t and $\overline{r_i}(t)$ is the smoothed throughput computed using exponential moving average. The flows being all independent (in terms of LMS channel evolution), all the throughput $r_i(t)$ are independent over time and thus different. The flow i^* elected to be transmitting is the one having the biggest $f_i(r)$ value, computed as follows:

$$i^* = argmax(f_i(r)) \tag{6.4}$$

To prevent flow sending blocking and to ensure fairness between the flows, the value $r_i(t)$ is divided by $\overline{r_i}(t)$. This ensures that a flow does not send during a too long time, starving the other flows and causing high latencies and jitter, which is not desired.

In our context, the throughput efficiency of the LMS link is directly linked to the channel capacity. Thus, this algorithm favors the flows having the best LMS channel capacity over time, decreasing the time needed by HARQ to decode each packet.

Proportional Fairness is designed to work with best-effort traffic such as TCP. However we show in Section 6.3 that adding a queue policy makes this scheduler compatible with VoIP traffic. This new scheduling policy is called **Controlled Delay Scheduler (CoDeS)**, and is detailed in Section 6.3.

Another sizing parameter that needs to be considered is the scheduler queues size. The storage capacity in a satellite is limited and for schedulers having as many queues as there are user's destinations, there is a need to optimize the buffer capacity. The authors of [18] give a formula to compute the buffer capacity. The total buffer capacity follows equation 6.5:

$$B = \frac{\overline{RTT} * C}{\sqrt{n}} \tag{6.5}$$

In this equation, $\overline{RTT}$ is the average Round-Trip Time of the flows, C is the bandwidth of the LMS link, and n is the number of flows sharing this link. This is a more accurate extension of the common rule $B = \overline{RTT} * C$. Thus, with PF,

each queue has a size of B/n in order to have a total capacity of B. The number of users n can easily be computed by the scheduler by identifying the number of destinations.

Finally, we compare in this paper PF scheduler performance with other schedulers and queuing policies that do not aim to optimize the LMS channel capacity:

- **Round Robin (RR):** each queue is served alternatively sending only one packet, then letting the other flows transmit;
- **DropTail Small buffer (DT_S):** only one queue with FIFO policy, the size of the queue is the same than in one queue of PF or RR, *i.e.* B/n. Thus, the global storage capacity is different from PF or RR;
- **DropTail High buffer (DT_H):** only one queue with FIFO policy, the size of the queue is B, meaning the same total capacity than PF or RR.

6.2.1 Implementation

The scheduling policies have been implemented in *ns*-2, improving the *Queue* and *PacketQueue* classes, particularly the *enque*() and *deque*() functions.

The *Queue* class implements the whole queue of a node. It stores the packets in the buffer and forwards them when the channel is available. It also implements queue management policies. A *Queue* uses one or more *PacketQueue* objects, representing the queue buffers. A *PacketQueue* aims to store the packets in one single linked list of packets. In our case for the LMS channel, we use one Queue representing the scheduler, using as many PacketQueue objects as there are buffers in the scheduler.

An overview of the *ns*-2 implementation of the schedulers in given in Figure 6.5. We can observe the *QueueScheduler* class using several *SchedulerPacketQueue* objects. Each of these *PacketQueue* computes the $f_i(r)$ value corresponding to the evaluation function associated to the flow. Then *QueueScheduler* chooses the best *SchedulerPacketQueue* to dequeue according to the policy implemented in the children classes: *RRScheduler*, *PFScheduler*, etc.

This makes this implementation generic and eases the addition of a new policy in the future.

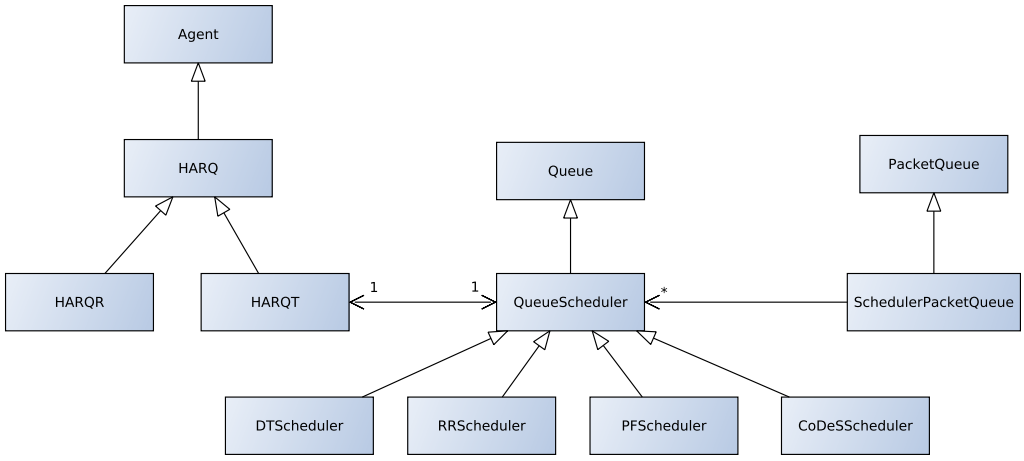


Figure 6.5 – UML Class Diagram of the *ns*-2 implementation

6.3 Scheduling VoIP users

We first study the impact of scheduling policies on the LMS channel, for VoIP traffic. Thus, this service, carried by UDP, is easier to study than services carried by TCP, and allows us to get firsts conclusions on the scheduling policies. The goal of the scheduling policies in this scenario is to maximize the Quality of Experience (QoE) of the users, which is mainly impacted by the latency, the jitter and the error rate.

The use of Call Admission Control (CAC) [75] is not suited for this scenario due to the high channel variability. These kinds of schemes allow each user to have a minimum capacity guaranteed, but this implies to monitor the channel to assess the worst-case channel capacity. However, the high channel variations cause attenuation up to 60 dB. Considering this worst case would imply a very low number of users to share the link, and an under-utilization of the LMS channel capacity.

We show in this section that using a Proportional Fairness (PF) scheduler conjointly with a queuing policy allows to greatly improve the LMS channel usage, with an efficiency close to the maximum measured in the previous Matlab simulations, in Section 6.1. We call this new scheduling policy **Controlled Delay Scheduler (CoDeS)** and detail its algorithms in the next sections.

6.3.1 Transport protocol and traffic

VoIP traffics [76], carried by UDP, have the characteristics given in Table 6.2. As we can see, our VoIP packets have a size of 210 bytes. However, our HARQ module

Property	Value
Rate	64 kb/s
Distribution	Pareto
Burst time	500 ms
Idle time	50 ms
Packets' size	210 bytes

Table 6.2 – Characteristics of VoIP traffic chosen

uses 1115 bytes packets which is larger than the VoIP packets size. Thus, every packet is filled in HARQ with zeros to get the correct size, leading to a maximum useful capacity of the LMS link of 9.4 Mb/s, instead of 50 Mb/s without padding. Nevertheless, other HARQ versions can use smaller packets, leading to a better optimization of the link capacity.

Following ITU-T G114 [19], the maximum acceptable delay for toll quality satellite links is 400 ms (note this value is a maximum, a fair compromise is to be around 200 ms). The jitter also needs to be minimized and the loss rate should not be higher than 3 %. These restrictions are mandatory to ensure a good Quality of Experience for the users. The challenge is to find a scheduling policy minimizing the losses and the jitter, while maximizing the throughput for each user. The QoE of the users is here measured through the Mean Opinion Score, which has been detailed in Chapter 3.

6.3.2 Schedulers tested

We carry out our simulations with several schedulers and queuing policies: DT_H, DT_S, RR, PF, and CoDeS later in this Section. We choose Proportional Fairness (PF) against M-LWDF, EXP-PF (although defined for real-time transmission) or generic algorithms such as BBS or BPS for several reasons:

- PF is a well-known and efficient scheduling system for fairness, simple to implement and computational efficient. Its only metric to optimize is the throughput of the LMS link, which directly leads to an improvement of HARQ performance;
- we demonstrate in the following that adding a simple queue management scheme to PF (CoDeS) allows to efficiently handle real-time traffic;
- the maximum gain measured with the first Matlab simulations is almost

reached with PF or CoDeS. Thus, there is no need to study other schedulers to improve performance, because the maximum is already reached. The only question concerning other schedulers deals with their ability to obtain similar results compared to CoDeS.

6.3.3 First results

In our simulations, we vary the number of parallel flows from 25 to 200, to simulate different loads on the LMS channel. With a low number of users, the buffers in the scheduler are never full nay empty, leading to a scheduling inefficiency. However, with a higher number of users, the throughput entering the LMS channel is higher than its capacity, causing drops, and assessing the impact of scheduling policies.

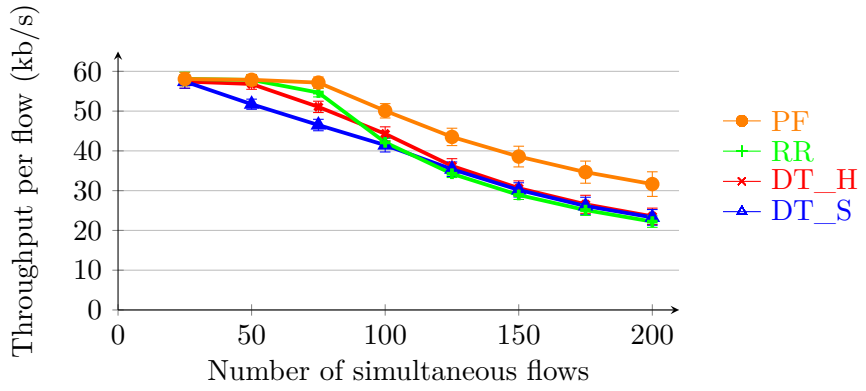


Figure 6.6 – Throughput obtained with different policies

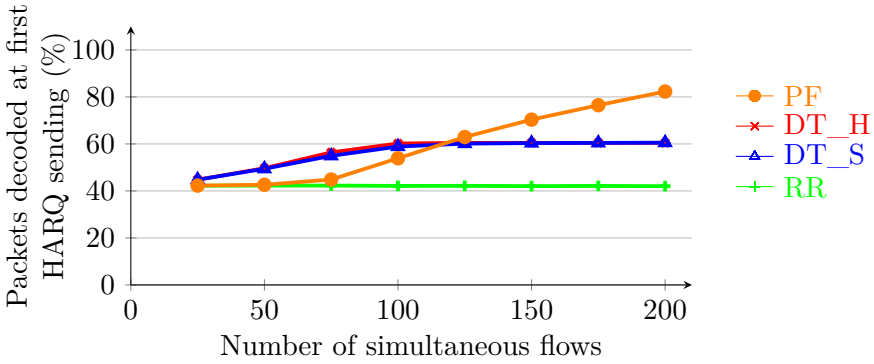


Figure 6.7 – Percentage of packets decoded at first HARQ sending with different policies

We first observe in Figure 6.6 that PF achieves a better throughput than the other scheduling policies, whatever the number of users. This good result is directly

linked to the optimization of HARQ, and thus the channel capacity, by PF, as seen in Figure 6.7. This scheduler decreases the average number of retransmissions needed to decode a packet when the channel is at full capacity. However, when the number of users is low and the LMS link is not at full capacity, PF does not achieve a better performance compared to the other policies due to the too low number of packets waiting in the queues: there is not enough packets in the queues to always find one with a good channel.

Moreover, even if more packets are decoded at first HARQ transmission with 75 users with DT compared to PF, the total number of packets decoded is higher with PF, due to the better success rate at the first and second retransmissions by HARQ. An overview of HARQ performance with 75 users is given in Table 6.3, where the higher number of retransmissions with PF and RR does not negatively impact on the transmission quality because the LMS channel is not saturated, and can handle more retransmissions.

	Proportion of packets decoded (%)				
#retransmissions	0	1	2	3	Drop
DT_S	50.11	23.16	14.65	12.07	1.21
DT_H	51.87	22.93	13.97	11.23	1.10
RR	42.36	24.28	24.71	8.65	0.61
PF	44.89	26.26	22.57	6.27	0.39

Table 6.3 – HARQ performance comparison with 75 users

A consequence of this good HARQ performance is a lower loss rate with PF compared to the other policies, as seen in Figure 6.8. As each packet needs less LMS channel capacity to be decoded, PF allows more packets to be transmitted on the channel, decreasing the queues backlog, and the queues drops. This Figure also shows that PF allows more users to transmit data before reaching the saturation point and a 3 % loss rate.

To comfort these results, we can also look at the spectral efficiency of the LMS channel, as shown in Figure 6.9. This figure represents the coded bits load on the LMS channel depending on the useful bits load. Each point of a curve represents a different number of parallel flows. We can see that PF performs a better channel usage, with up to 6 Mb/s useful bit load for 9 Mb/s coded bits load, whereas other policies cannot achieve a useful bits load higher than 5 Mb/s. Thus, for a same number of coded bits, PF can transmit more useful bits (meaning less redundancy bits) and the LMS channel is optimized with this scheduling policy.

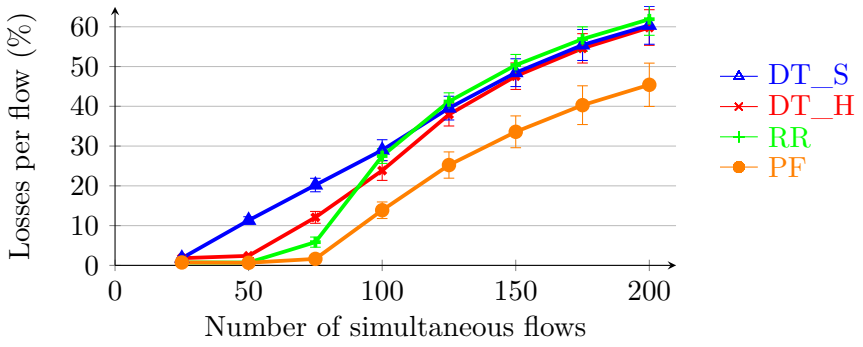


Figure 6.8 – Losses obtained with different policies

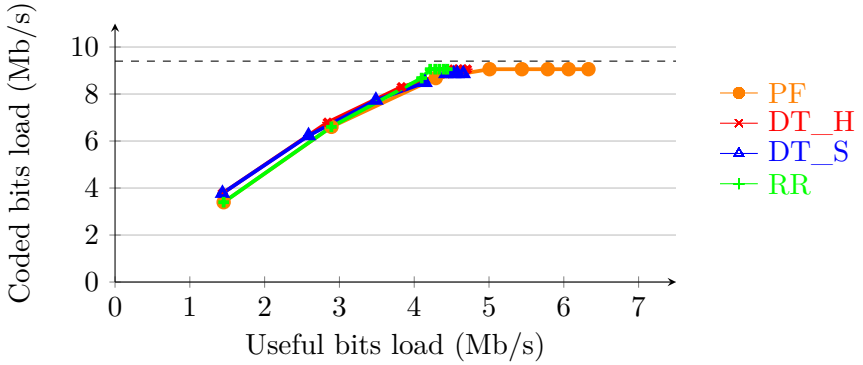


Figure 6.9 – LMS channel spectral efficiency

We can also have a look to other metrics which are taken into account in the users QoE. We observe in Figure 6.10, showing the transmission delays, that RR and DT_H induce a high delay in the transmission, which can be higher than 300 ms, whereas PF and DT_S achieves lower delays, which are in acceptable ranges, taking into account the satellite environment context.

The good PF performance is however mitigated by a high jitter, as shown in Figure 6.11. This high jitter with PF is due to the unknown waiting time of the packets in the buffer, which depends on the evolution of the LMS channel. This varying waiting time directly impacts the jitter, which is in range that are not compatible with a good QoE for the VoIP users [30]. This high jitter needs to be lowered if we want to achieve a good QoE with PF.

6.3.4 Analysis

In this section we compare each policy in terms of queue size, latency and Quality of Experience. We compute the Mean Opinion Score (MOS) to assess user satisfaction.

We first we observe that the maximum capacity of the LMS channel is reached

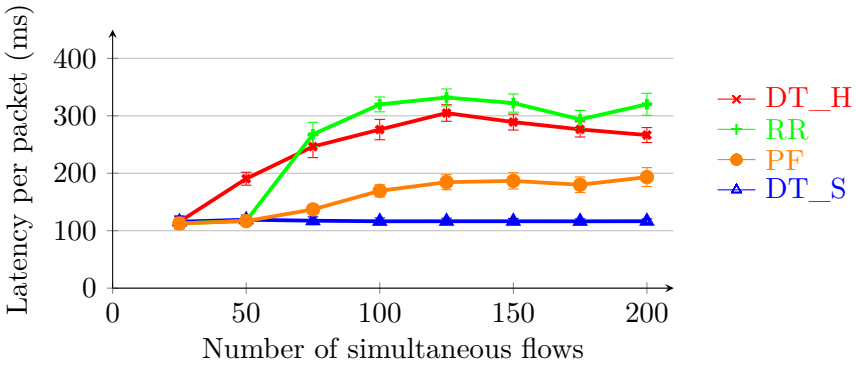


Figure 6.10 – Latency obtained with different policies

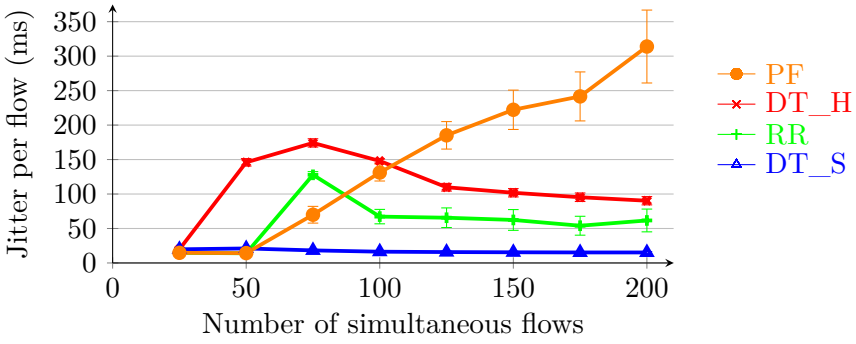


Figure 6.11 – Jitter obtained with different policies

for a number of users between 75 and 100. This number is a bit higher with PF compared to the other policies due to the better performance of HARQ. For higher number of users, the number of losses, due to HARQ or queue overflow, becomes too high to ensure a good QoE for the users. Thus, we consider in this scenario the maximum number of parallel users allowed in the network as 100 users.

A first overview of all the policies metrics show that RR and DT_H result in poor performance. The high buffer capacities induce a high delay for both policies: when the buffers (having a capacity of B packets) are full, the waiting time of a packet entering the queue is B times the transmission time of a packet, including HARQ retransmissions. This high latency has to be added to the fact that HARQ performance is not optimized with these schedulers compared to PF. As a conclusion, RR and DT_H are bad candidates for our scenario.

The two remaining candidates are DT_S and PF. At a first sight, DT_S may appear a good candidate due to the low jitter and latencies obtained. However, DT_S is penalized by a high error rate, even for a low number of flows, due to the low buffer capacity. As seen in Figure 6.8, with 75 users, DT_S has a loss rate

of 20 %, which is not suitable for a good QoE, unlike PF which has a loss rate of only 3 % which remains acceptable. Moreover, another disadvantage of DT_S is the absence of HARQ optimization, as presented in Table 6.3, leading to a high LMS capacity usage to decode each packet.

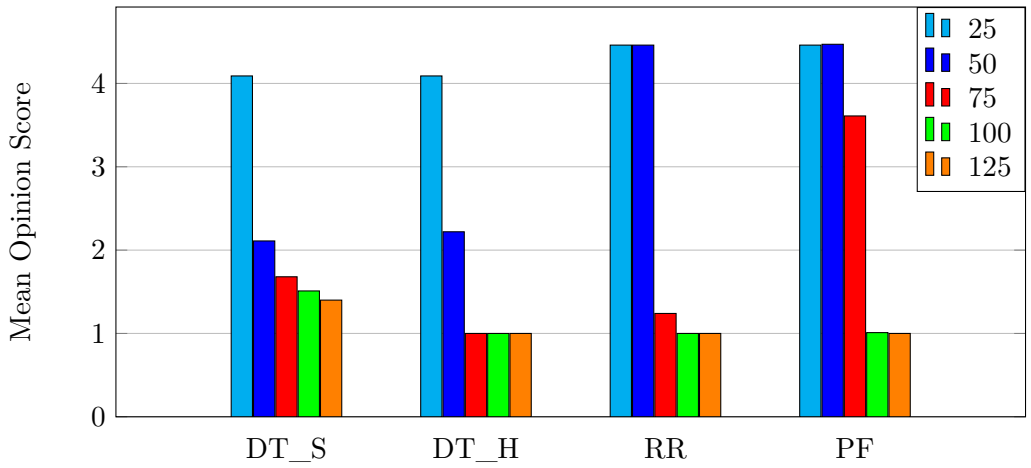


Figure 6.12 – Mean Opinion Score of scheduling policies as a function of the number of flows

Finally users QoE with all scheduling policies is given through the Mean Opinion Score, and shown in Figure 6.12. We observe that PF achieves the best MOS when the saturation point is not reached (*i.e.* less that 75 users). The high loss rate highly penalizes DT_S performance, leading to a low QoE. In the context of satellite communications, the MOS is impacted by higher delays compared to terrestrial transmissions, and the users are more tolerant concerning their QoE. This is taken into account in the MOS computation (following ITU-T G114 [19]) by using an Advantage Factor, set for satellite users to 20. We can consider a good MOS value when it is higher than 3, which is the case for PF when the LMS channel is not saturated.

PF is then the scheduling policy performing the best performance, but at a price of a high jitter. The LMS channel evolution makes the waiting time of the packets totally unpredictable, causing this high jitter. In the following, we show that adding a queuing policy to PF greatly decreases the jitter without impacting on other metrics.

6.3.5 Improving Proportional Fairness performance

We propose to improve PF performance by adding a queue management scheme to the PF scheduler. We note that the high waiting time of the packets in the queues can penalize VoIP transmission: if a VoIP packet waits a too long time in the buffer, at reception at the receiver, it may be obsolete and not useful for the application layer, and discarded. Moreover, when a queue is full, the actual policy is DropTail, meaning that the incoming packet is directly dropped, even if it could be transmitted quickly in case of good LMS channel capacity.

To summarize, we have:

- packets which remain in the buffer but are discarded by the application layer because they are obsolete;
- packets dropped due to buffer overflow that could be still exploitable by the application layer.

This observation leads us to introduce a new queuing policy conjointly to PF, called Controlled Delay Scheduler (CoDeS). The aim of this policy is to set a time threshold in the queues beyond which the packets are dropped. This queuing policy then drops the packets remaining during a too long time in the buffers, instead of using a DropTail policy in case of buffer overflow. The average waiting time of the packets is then lower, as well as the jitter.

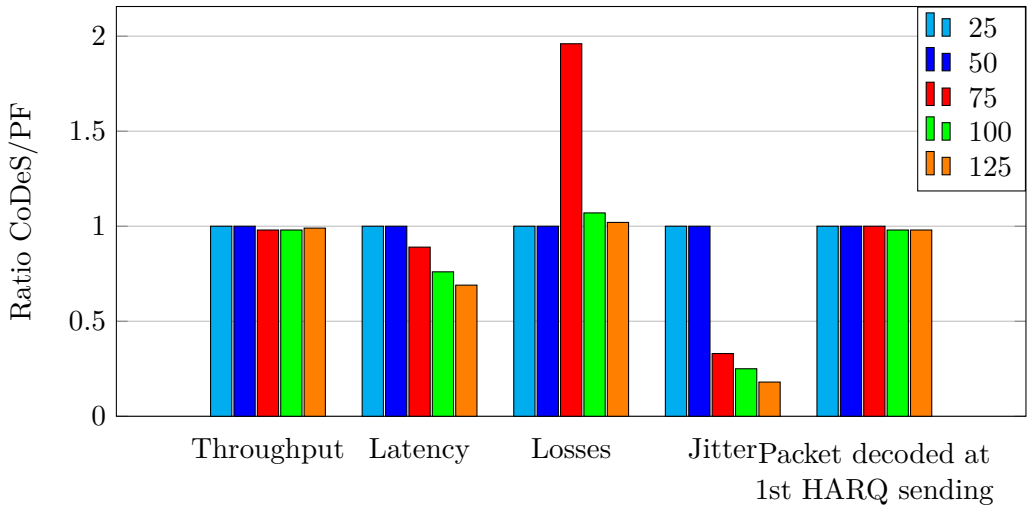


Figure 6.13 – Comparison of the ratio of performance between CoDeS and PF, with a timeout of 100 ms

The impact of CoDeS compared to PF on QoS metrics is summarized in Figure 6.13, where the timeout is the queue has been set to 100 ms. We observe as expected a decrease of the latency and mostly the jitter, without impacting the other metrics. The number of losses is approximately the same than with PF, with acceptable values up to 100 users.

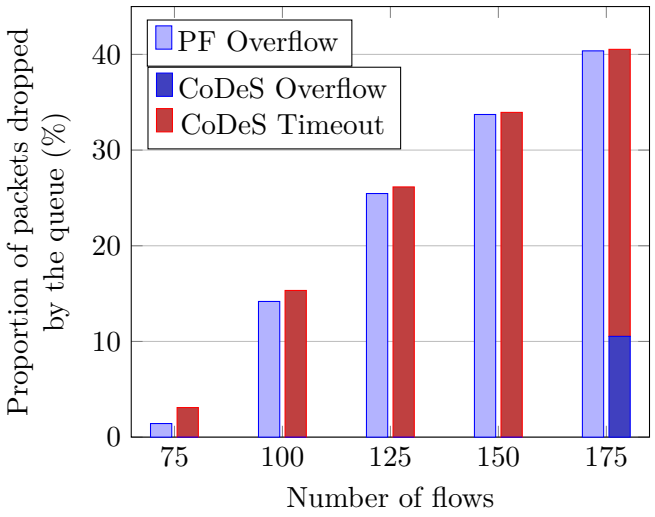


Figure 6.14 – Comparison of the cause of drops with PF and CoDeS, with a timeout of 100 ms

Figure 6.14, comparing the cause of the drops in the queues between PF and CoDeS, show that the total number of drops does not increase significantly with CoDeS, but the packets are now mainly dropped because of timeout and no more because of buffer overflow. This behavior change induces a lower sojourn time of the packets in the queues, leading to a lower latency and jitter with CoDeS, as shown in Figure 6.15. In this figure, we can see the important jitter decrease. This drop is higher when the timeout value is low, with a jitter value reaching the same rates than DT_S for small timeout values.

However, CoDeS has no impact on the loss rate and on the LMS channel optimization. Then the value of 100 users is still the limit beyond which the loss rate is too high to ensure a good QoE. The only QoE improvement is then brought by the lower latency and jitter.

The CoDeS timeout value in the queues now needs to be set in order to optimize the policy efficiency. Lower timeout values induce low latencies and jitter, but at a price of a higher loss rate. We observed that the minimum jitter and latency values are reached for timeout values of 50 ms, as shown in Figure 6.15. However, as seen in Figure 6.16, the loss rate tends to increase with values lower than 50 ms. This low

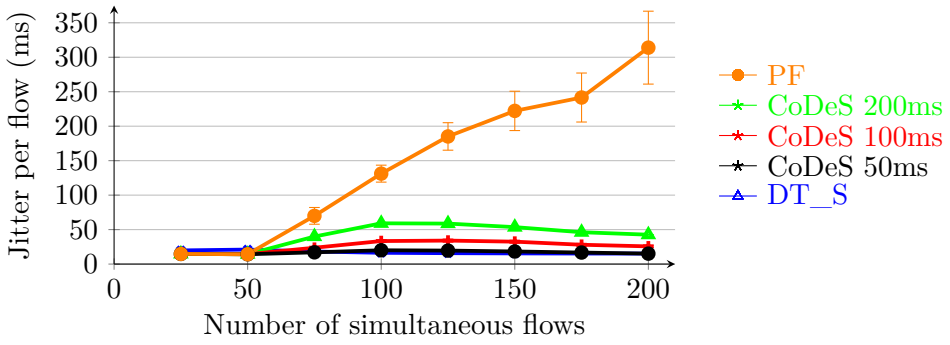


Figure 6.15 – Jitter obtained with different policies

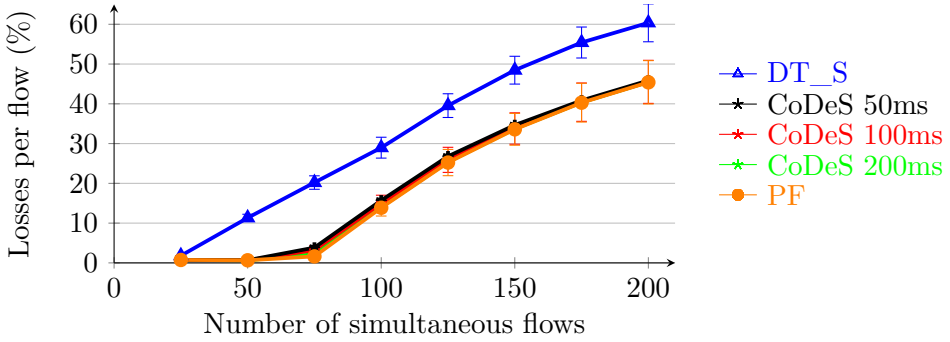


Figure 6.16 – Losses obtained with different policies

timeout value causes the packets to be dropped too early, leading to empty buffers and a bad efficiency of the policy. Thus, a correct value of timeout should be set between 50 and 100 ms, to have the best compromise between jitter and losses.

Finally, we can observe in Figure 6.17 that CoDeS improves the MOS compared to PF, with a timeout value of 100 ms. Thus, combining PF with a queuing policy, such as CoDeS, improves the users QoE, by lowering the jitter and the latency, and optimizing the LMS channel capacity.

The results with VoIP and UDP show that CoDeS maximizes the users QoE. However, the impact of this scheduler can be tested with other transport protocols such as TCP, where high jitter, latency and loss rate highly impact TCP performance. This is the goal of the study of the following Section 6.4.

6.4 Scheduling TCP users

We have seen in the previous section that in the case of VoIP traffic and UDP protocol, CoDeS obtains good results in terms of LMS channel capacity optimization and QoE of the users. We now study the impact of the schedulers detailed before

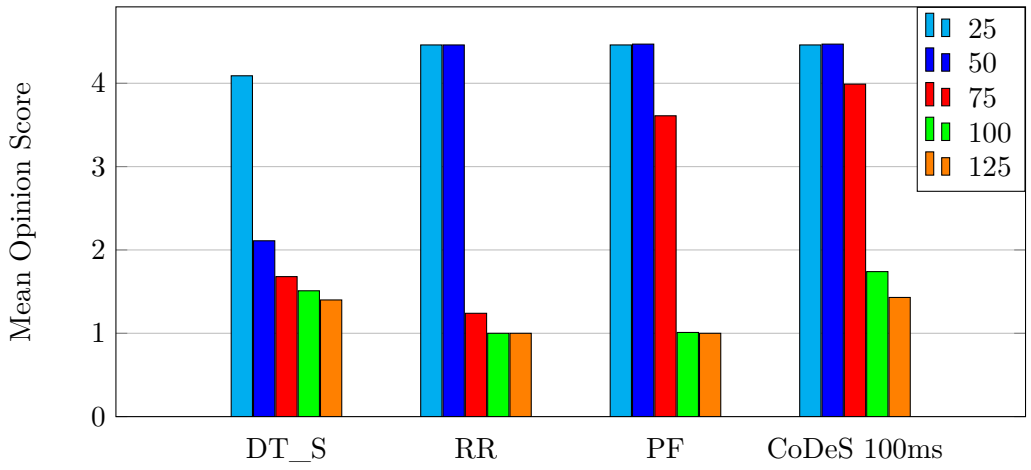


Figure 6.17 – Mean Opinion Score of scheduling policies as a function of the number of flows

in the case of best-effort traffic, carried out by TCP, and assess if our previous conclusions are still valid.

6.4.1 Scenarios

In the case of best-effort traffic carried out by a TCP transmission, the objectives are now:

1. a good delivery of every packets without any error;
2. to maximize the transmission goodput for each user.

In that case, the schedulers must ensure a transmission as fast as possible of every packet on the LMS channel while avoiding out-of-order packets and TCP timeout.

Concerning the schedulers chosen, we test the same scheduling policies than in Section 6.2. We keep DT_H, DT_S and RR as comparison basis, and study the impact of PF and CoDeS:

- PF is totally suited of best-effort traffic, thus we study its impact on TCP performance compared to the other policies;
- CoDeS adds a drop policy in the queues based on a timeout. We need to study whether this scheme is still relevant in the case of TCP traffic.

These schedulers are tested with different numbers of parallel flows, simulating different loads. We also enable and disable the reordering mechanism to study its

impact. The TCP variants used are still TCP NewReno and CUBIC as in the previous chapters.

6.4.2 Impact of scheduling on TCP without reordering

We plotted the goodput gain achieved, compared to DT_H (a simple FIFO queue with the same buffer capacity than PF or RR), for each TCP user, depending on the number of parallel flows, for TCP NewReno in Figure 6.18 and CUBIC in Figure 6.19. We observe that PF obtains the best improvements in both cases, as well as Round Robin, which even sometimes beats PF when a low number of users are sharing the network. This result seems counter intuitive as RR does not optimize the LMS channel capacity usage, unlike PF, and is explained later.

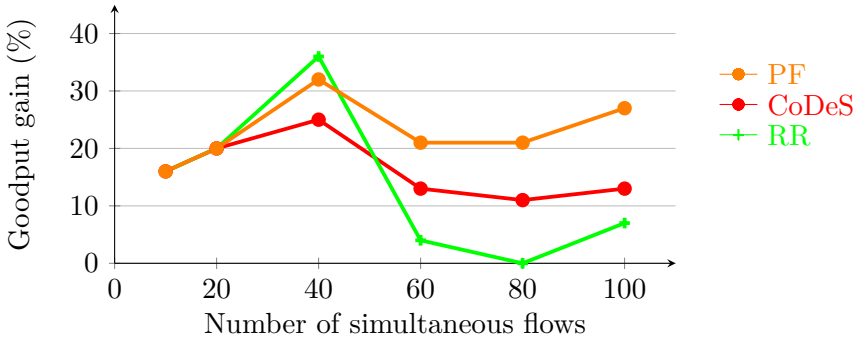


Figure 6.18 – Goodput gain per user for different scheduling policies compared to DT_H (NewReno)

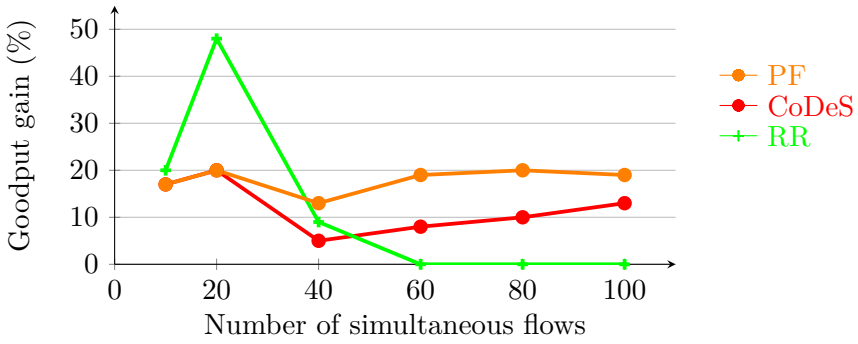


Figure 6.19 – Goodput gain per user for different scheduling policies compared to DT_H (CUBIC)

In Figure 6.20, showing the total goodput in the network, we can also see that PF obtains the best performance, with up to 20 Mb/s of useful bits.

We also observe in these Figures that CoDeS does not improve TCP performance. On the contrary, TCP goodput is worse with CoDeS than PF. This low

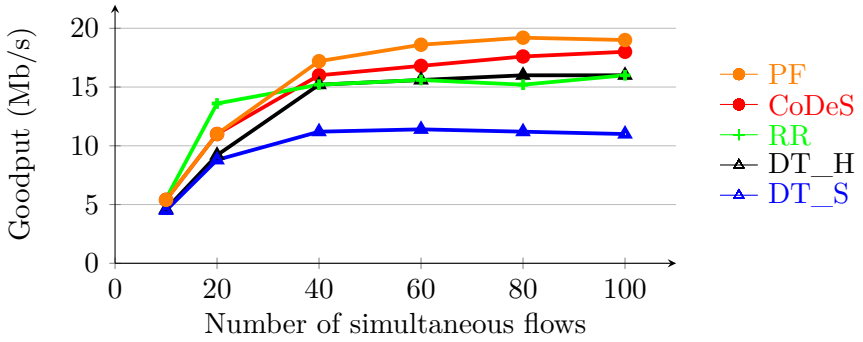


Figure 6.20 – Global goodput for different scheduling policies (CUBIC)

performance has several reasons:

- TCP congestion control algorithm adapts its sending rate to the network capacity, and reacts to congestion, via DUPACK or RTO. This behavior prevents TCP to uselessly overflow the buffers, reducing the number of packets dropped by the buffers compared to UDP, when a lot of users are transmitting. Thus, implementing a new queue management is useless or counterproductive;
- TCP is a reliable protocol, unlike UDP, and the transmission time is now less important than in Section 6.3. Dropping a packet in the queue results in a loss detection by TCP, a retransmission and a congestion window decrease because TCP assumes there is congestion. It is more efficient now to delay the packets, at the price of more RTO, letting the transport layer adapt itself to the potential timeouts.

Thus CoDeS is not suited for TCP transmissions, and PF obtains better results, while less complex.

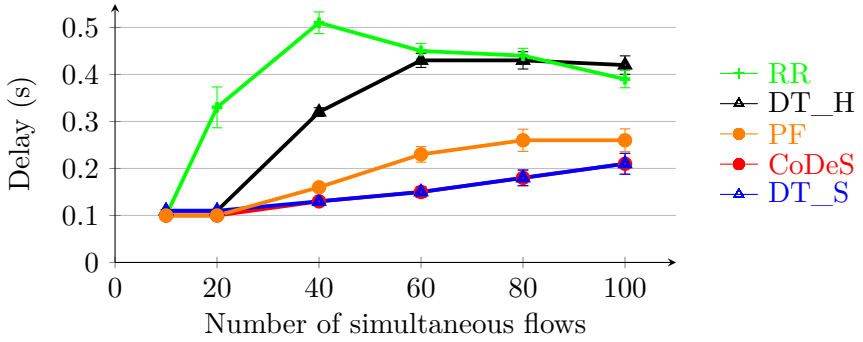


Figure 6.21 – Transmission time per packet (CUBIC)

When looking at the other metrics, we first observe the transmission time is

still very important with RR and DT_H, as seen in Figure 6.21. The transmission delay per packet, taking into account potential TCP retransmissions (out-of-order delivery), can reach up to 500 ms. However, these long delays do not impact TCP performance and do not generate spurious timeout as the RTO adapts to the measured RTT. On the other hand, DT_S, CoDeS and PF obtains low transmission delays.

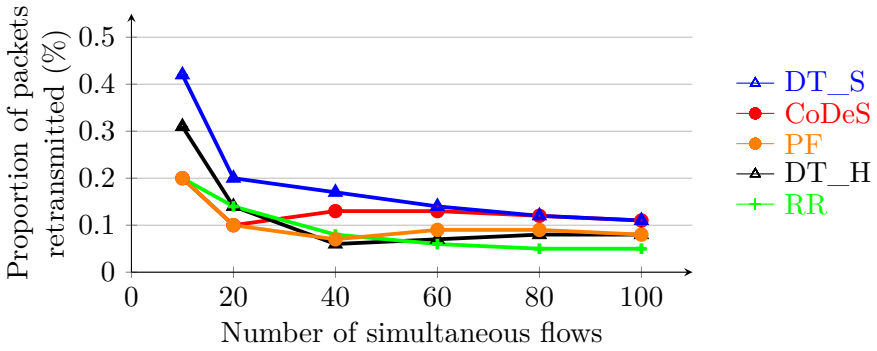


Figure 6.22 – Proportion of packets retransmitted per flow (CUBIC)

Finally, in Figure 6.22, we can see the results concerning the proportion of retransmissions by TCP, meaning the number of packets retransmitted divided by the total number of packets sent. We can observe that PF, RR and DT_H have the lowest retransmission rate. CoDeS has a number of retransmission higher due to the queue drops by timeout, and DT_S has the highest rate due to the lower buffer capacity.

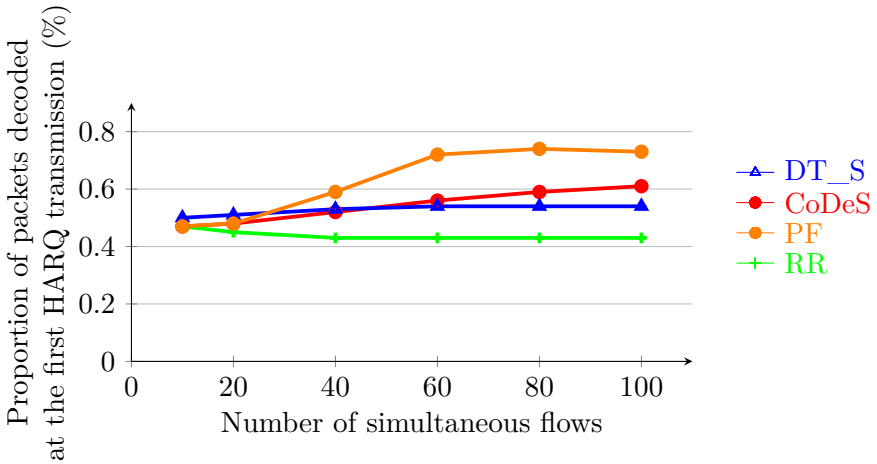


Figure 6.23 – Proportion of packets decoded at the first HARQ transmission (CUBIC)

We have seen in the results that RR obtains good goodput for a low number of users without optimizing the LMS channel capacity, as shown in Figure 6.23. The reason of this good performance is the spacing between each transmission in a same flow on the LMS channel, compared to PF which has a bursty distribution. This spacing mitigates the number of out-of-order packets received by TCP, and then the number of DUPACK and retransmissions.

We can analytically assess the impact of this spacing: considering the decoding probability depending on the number of retransmissions, we can compute the average number of bits sent per packet $\overline{len}$:

$$\overline{len} = 16890\,bits \tag{6.6}$$

We also know the bandwidth of the LMS channel (50 Mb/s), thus we can compute the average time needed to send a packet:

$$\bar{t} = \frac{\overline{len}}{BW} = 0.34\,ms \tag{6.7}$$

Finally, when n users are transmitting, each flow sends a packet every $n * \bar{t}$ seconds. In case of additional time Δt needed to decode a packet (mainly because of HARQ), TCP triggers a retransmission by DUPACK if:

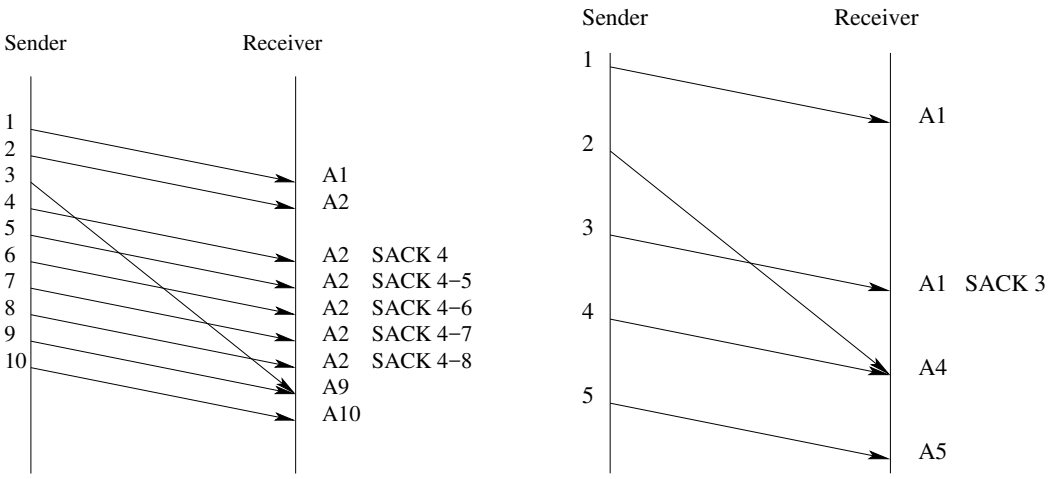
$$\Delta t \geq 3n\bar{t} \tag{6.8}$$

We know Δt is between 15 ms and 20 ms, we can then compute the number of users beyond which a DUPACK retransmission is triggered:

$$n \leq \frac{\Delta t}{3\bar{t}} \approx 19\,users \tag{6.9}$$

Thus, if we have more than 20 users transmitting, the spacing in the transmission mitigates the impact of DUPACK, improving TCP performance, and explaining the inflection point with RR around 20 users in Figure 6.19. Figure 6.24, illustrates this behavior.

Moreover, even if a loss is detected by DUPACK, the number of packets to be retransmitted is lower with RR, leading to less capacity on the LMS channel used for spurious retransmissions. Figure 6.25, comparing the goodput achieved by RR with and without reordering, confirms that with more than 20 users, the goodput achieved by this scheduler without reordering equals the goodput with reordering. Thus, the impact of out-of-order packets is totally mitigated by RR beyond this number of users.



With low spacing: loss detected

With high spacing: no loss detected

Figure 6.24 – Impact of packet spacing on loss detection by DUPACK

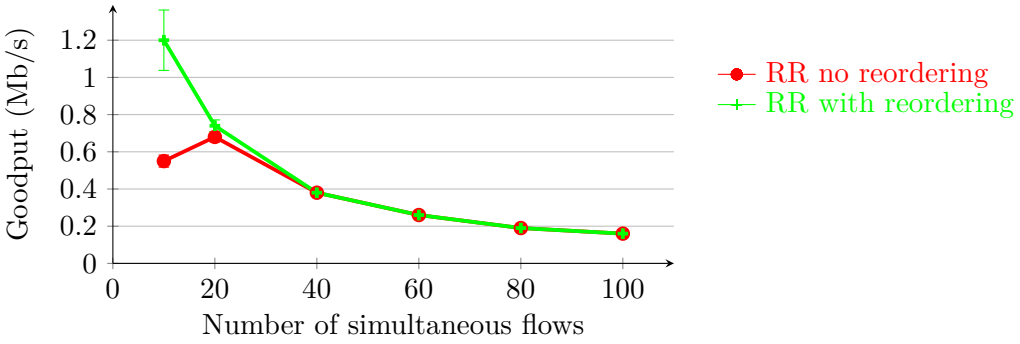


Figure 6.25 – Performance comparison of RR with and without reordering (CUBIC)

Finally, this low number of DUPACK and retransmissions with RR compensates the absence of LMS channel optimization, and RR shows performance similar to PF for a low number of users. These two schedulers can be considered as good candidates for TCP traffic, when no reordering mechanism is present conjointly with HARQ. However, with a high number of users (more than 60), PF obtains better results than RR: the channel optimization with PF becomes more significant on TCP performance than out-of-order packets mitigation with RR.

However, we saw that CoDeS does not improve TCP performance, and can be counterproductive: it does not add new mechanisms in addition to TCP congestion control, and packets drops are now very disadvantageous for TCP flows.

These results have been obtained without the reordering mechanism presented in Chapter 4, and we can now assess the impact of these policies when the reordering mechanism is enabled.

6.4.3 Impact of scheduling on TCP with reordering

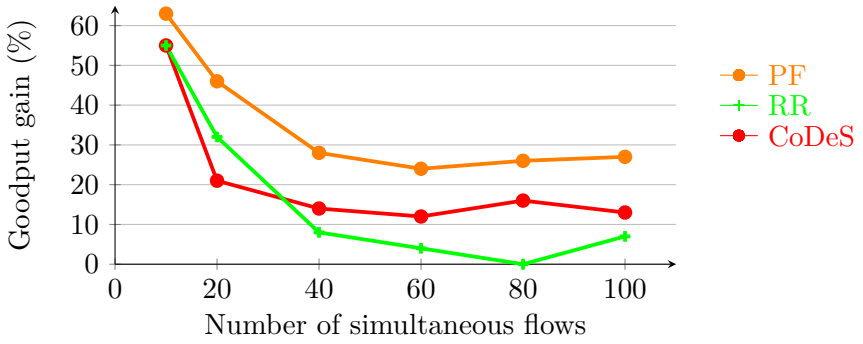


Figure 6.26 – Goodput gain per user for different scheduling policies compared to DT_H (NewReno)

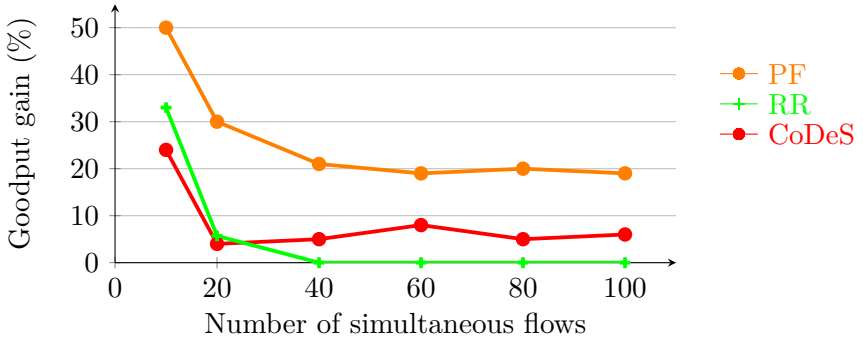


Figure 6.27 – Goodput gain per user for different scheduling policies compared to DT_H (CUBIC)

TCP goodput gain compared to DT_H, for each scheduling policy and when the reordering mechanism is enabled, is presented in Figures 6.26 (TCP NewReno) and 6.27 (CUBIC). We can see that PF obtains the best goodput gain for the users, and that RR is not competitive anymore. Thus, the advantages brought by RR to deal with out-of-order packets are useless here, as the reordering mechanism mitigates the number of out-of-order packets, and only PF optimizes the LMS channel capacity.

We can observe in Figure 6.28 that PF is still achieving the best total goodput in the network compared to the other policies. DT_S cannot obtain good performance because of its low buffer capacity, and DT_H has results similar to RR due to the absence of channel optimization.

Concerning the other metrics, we see in Figure 6.29 that RR and DT_H have still a long transmission delay, directly due to their algorithm and large buffer

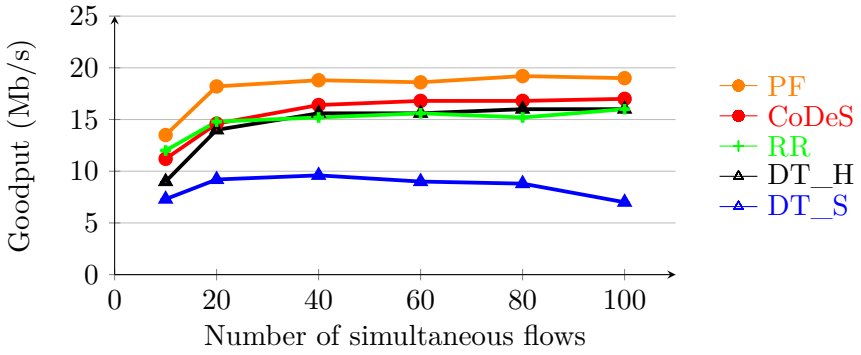


Figure 6.28 – Global goodput for different scheduling policies (CUBIC)

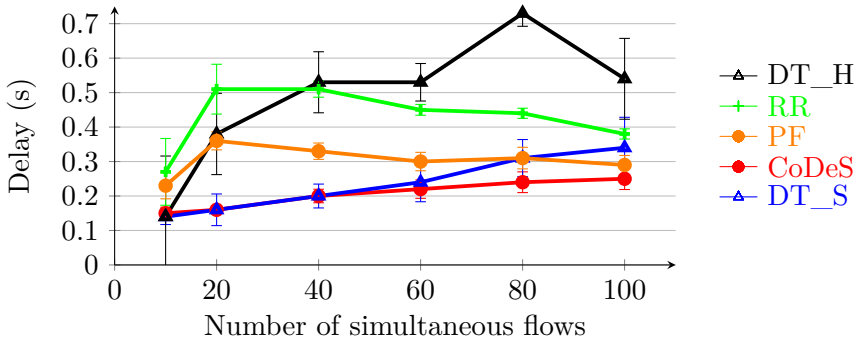


Figure 6.29 – Transmission time per packet (CUBIC)

capacity, while DT_S and CoDeS shows the lowest delays, due to the first to the low queue size and for the second to the timeout policy in the queues. With PF, the delay is around 300 ms, which remains an acceptable value for best-effort traffic taking into account the use of a satellite constellation and the presence of reliability schemes.

The proportion of retransmissions is presented in Figure 6.30, showing a similar proportion of retransmissions between RR and PF. However, whatever the scheduling policy, the proportion of retransmissions is higher compared to the previous section (Figure 6.22), when no reordering mechanism is enabled. This increase is mainly due to two main reasons:

- the reordering mechanism adds an additional delay to an already important transmission delays, increasing the number of TCP timeout and thus the number of retransmissions and spurious retransmissions, mainly for a low number of users, as shown in Figure 6.31 for the PF policy. This behavior can also be seen with RR;
- the main part of these retransmissions are due to congestion, corresponding

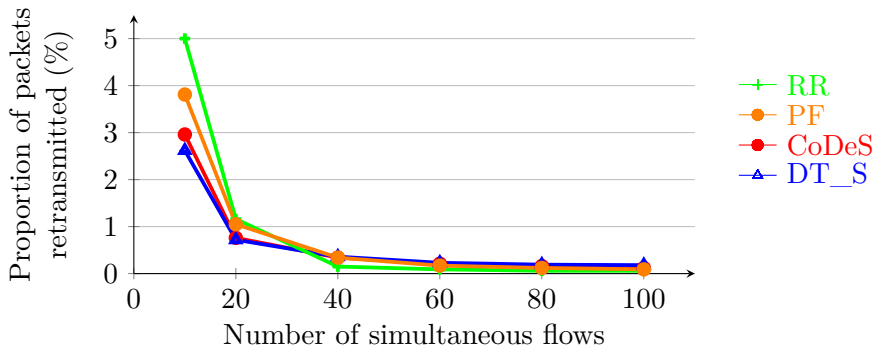


Figure 6.30 – Proportion of packets retransmitted per flow (CUBIC)

to normal behavior of TCP. These are congestion losses, and TCP is designed to manage these kind of losses. This is useless to deal with the queues size, the buffer overflow would still occur and it is better to optimize the storage capacity, limited in a satellite environment.

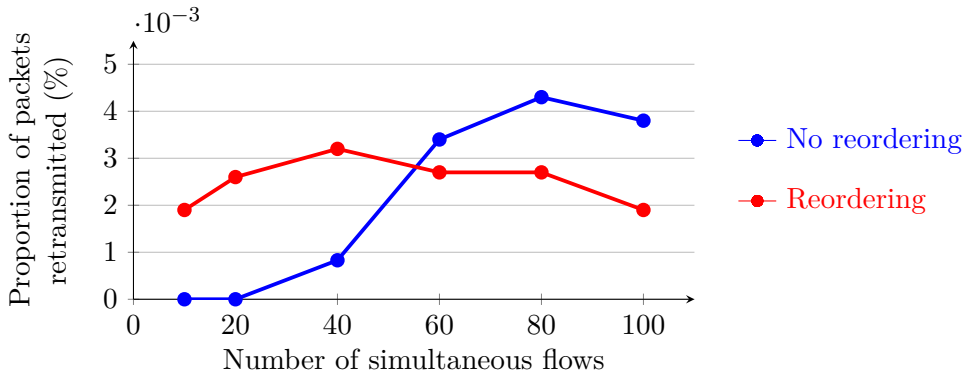


Figure 6.31 – Proportion of RTO events per flow (CUBIC, PF)

As for the scenario without reordering, CoDeS does not show any improvement in TCP performance, and is not suited for best-effort traffic. PF remains the only scheduler optimizing the transmissions with and without reordering mechanism, due to its LMS channel optimization.

We can also note that once reaching the saturation point, the goodput per user is similar regardless of the TCP variant used or the activation of the reordering mechanism. Thus, once there are enough users to always have packets in the LMS channel queues, the congestion control algorithms do not significantly impact on the performance, because TCP performance is driven by the performance of the scheduling policy. However, when the saturation point is not reached, we can see a significant improvement with CUBIC compared to NewReno, and with the

reordering mechanism, as seen in Figure 6.32.

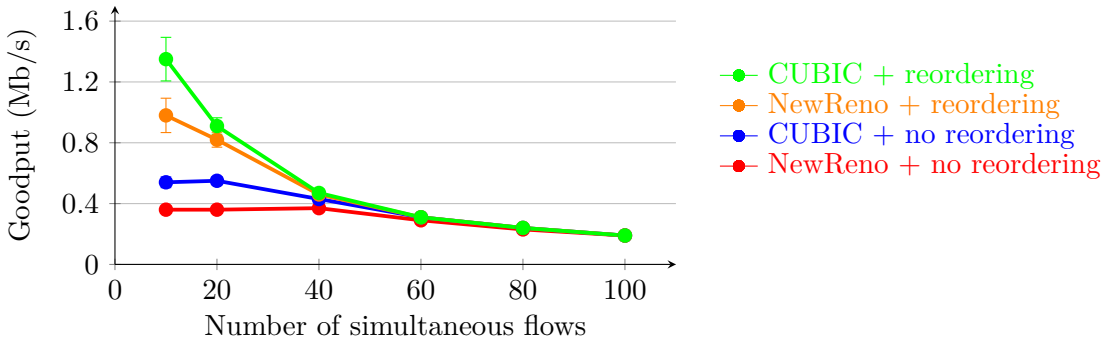


Figure 6.32 – Impact of transport protocol and reordering on TCP performance, using PF

6.4.4 Fairness between users

In the previous sections, we presented the mean results for all flows, but without giving any information on the fairness between these flows. A bad fairness would be prejudicial for the global performance and Quality of Service, as some users would have a very low goodput compared to the others, if not a lower goodput than it could have had with other scheduling policies.

In this section, we study the fairness between the users. We cannot use the Jain's fairness index [77], because of the varying LMS channel capacity. We use instead the minimal and maximal goodput values, as well as the Standard Deviation to compare the fairness between users.

Users	Min			Max			Mean			SD		
	PF	DT_S	RR	PF	DT_S	RR	PF	DT_S	RR	PF	DT_S	RR
10	0.44	0.43	0.46	0.58	0.51	0.61	0.54	0.45	0.55	0.035	0.014	0.033
20	0.45	0.41	0.56	0.60	0.47	0.72	0.55	0.44	0.68	0.028	0.012	0.027
40	0.38	0.24	0.37	0.45	0.32	0.41	0.42	0.28	0.38	0.016	0.016	0.004
60	0.27	0.15	0.25	0.35	0.21	0.27	0.31	0.19	0.26	0.017	0.014	0.003
80	0.21	0.10	0.19	0.27	0.17	0.21	0.24	0.14	0.19	0.015	0.014	0.004
100	0.16	0.07	0.15	0.23	0.14	0.17	0.19	0.11	0.16	0.017	0.012	0.004

Table 6.4 – Statistics on the goodput between DT_S, PF and RR without reordering

Table 6.4 shows these values for DT_S and PF and RR when the reordering mechanism is disabled. We can observe that the Standard Deviation is the lowest with RR, which could be expected due to the policy that does not favor one flow compared to the others. However, PF shows a larger Standard Deviation, but the

flow with the minimal goodput has almost always a better performance than the best flow with DT_S or RR (the flow with the maximum goodput).

This means that even if there is less fairness between the flows with PF compared to RR or DT_S, all the flows have a better performance with PF compared to the other policies. The difference between the best flow and the worst flow is between 10 % and 40 %, depending on the number of users sharing the LMS channel.

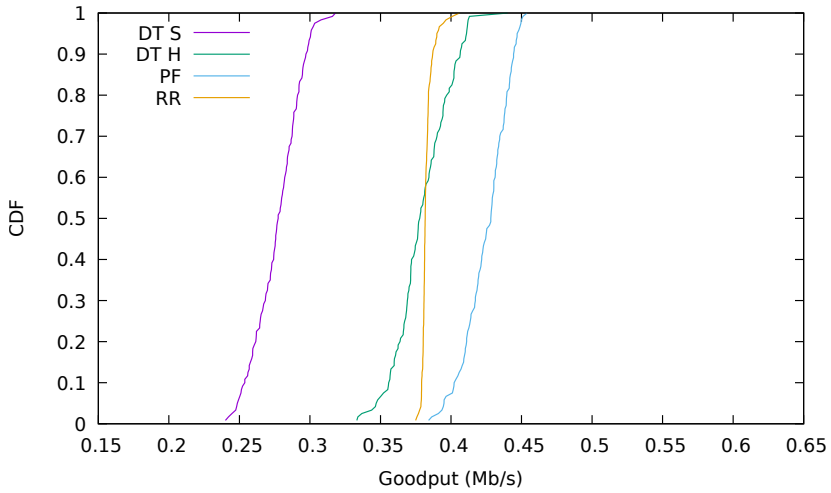


Figure 6.33 – CDF of goodput obtained by the users, with different scheduling policies (no reordering, 40 users)

These results can be summarized in Figure 6.33, showing the CDF of the different policies, with 40 parallel users. We can observe the good performance of PF compared to the other policies, and also the good fairness of RR between its flows.

Users	Min			Max			Mean			SD		
	PF	DT_S	RR	PF	DT_S	RR	PF	DT_S	RR	PF	DT_S	RR
10	0.85	0.60	0.77	1.62	0.84	1.45	1.35	0.73	1.20	0.143	0.056	0.162
20	0.75	0.25	0.61	1.01	0.53	0.78	0.91	0.46	0.74	0.054	0.041	0.032
40	0.41	0.18	0.36	0.54	0.28	0.40	0.47	0.24	0.38	0.026	0.020	0.004
60	0.27	0.07	0.25	0.35	0.20	0.27	0.31	0.15	0.26	0.020	0.020	0.005
80	0.20	0.12	0.19	0.28	0.16	0.21	0.24	0.11	0.19	0.018	0.020	0.004
100	0.16	0.03	0.15	0.22	0.12	0.17	0.19	0.08	0.16	0.013	0.015	0.004

Table 6.5 – Statistics on the goodput between DT_S, PF and RR with reordering

This conclusion still holds when the reordering mechanism is enabled, as shown in Table 6.5: even if the Standard Deviation is higher with PF, its performance with the worst flow is still higher than any flow from the other policies. These results are confirmed with the CDF presented in Figure 6.34.

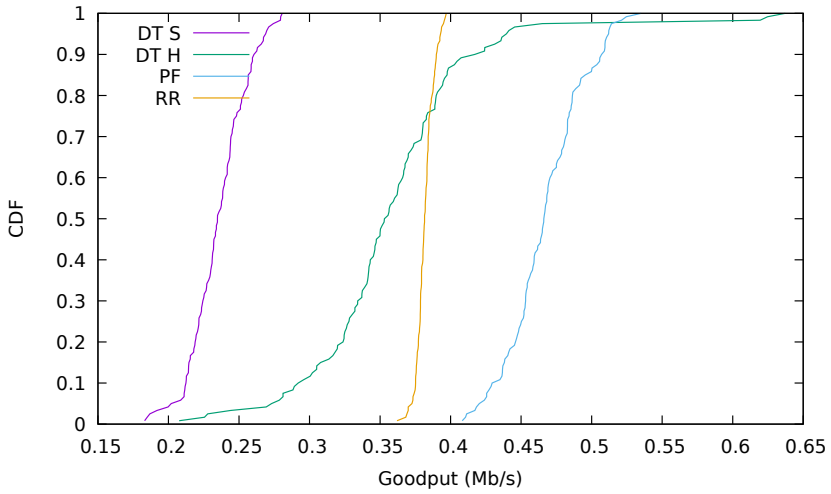


Figure 6.34 – CDF of goodput obtained by the users, with different scheduling policies (reordering, 40 users)

6.5 Conclusion

We highlighted in this chapter the performance gain brought by an efficient scheduling policy on the LMS link, optimizing the channel capacity.

In the case of VoIP traffic, we showed that a new scheduling policy, called CoDeS, which combines a Proportional Fairness scheduling policy to a queue management, allows to improve the Quality of Experience of the users. This policy optimizes the LMS channel capacity, with performance close to the maximum that can be expected, while mitigating the jitter.

In the case of TCP best-effort traffic, we showed that CoDeS is not useful anymore, and that Proportional Fairness is now the best scheduler for this context. It optimizes the LMS channel capacity, while maximizing the TCP goodput per user, with and without reordering mechanism. When the reordering mechanism is disabled, we also noticed and explained the good performance of Round Robin, for a low number of parallel users.

We can also note that PF is suited for other environments involving channel capacity variations, for example in 5G networks [78, 79]. As for LEO satellite constellations, the use of PF allows to increase the channel spectral efficiency by favoring the users with the best channel conditions, while providing fairness between the users.

Chapter 7

Conclusion

This thesis highlighted the importance of the impact of low layer reliability mechanisms on the transport layer performance. In the context of LEO satellite constellations, the variations of link delays, caused by the topology and the use of reliability schemes on the LMS channel such as HARQ, generate out-of-order packets that are misinterpreted by TCP, halving its congestion window, generating spurious retransmissions and then decreasing TCP performance.

After understanding and explaining the impact of HARQ on TCP performance, we first proposed to conjointly add with HARQ a reordering mechanism mitigating the impact of out-of-order packets on TCP. This new scheme highly improves TCP performance, whatever the variant used. We set the parameters of this mechanism to optimize TCP performance. We then analyzed the impact of some TCP options in this scenario, and showed that nearly all of the parameters tested did not have a significant impact on TCP performance.

However, unlike other TCP options, we emphasized the impact of DelAck as this TCP option either significantly increases or decreases TCP performance depending on both the TCP variant and the reordering mechanism activation. The measurements driven allowed to conclude that the pacing introduced by DelAck is counterproductive when a high number of retransmission occurs. The additional delay brought out by this option also increases the probability of TCP timeout, which is an issue considering long delays occurring in this environment.

Finally, we explored the feasibility of adding a scheduling policy on the LMS channel to optimize its capacity when several users are sharing the link. This led to the introduction of a new policy called CoDeS suited for VoIP traffic, which significantly improves the Quality of Experience of the users. In the case of best-effort traffic, we showed that a Proportional Fairness scheduler alone optimizes the channel capacity, at a low computation cost. We also highlighted the good performance of Round Robin, in some scenarios, as it mitigates the impact of out-of-order packets.

All these results allow to improve QoE when using a LEO satellite constellation: a better goodput for the users when using best-effort traffic and a better

Mean Opinion Score in the case of VoIP traffic. Conclusions applied to the context of satellite constellation can also be extended to wireless terrestrial cases, when reliability schemes induce out-of-order packets.

There is room for improvements and perspectives, the results of this thesis and recent research lead to future considerations that can further improve LEO satellite transmissions efficiency:

- another performance evaluation can be envisioned with new congestion control such as BBR or TCP enhancement such as RACK. Both schemes can be suited for space transmissions as they use different loss detection algorithms, that do not react similarly to the ones in classic TCP variants to losses;
- we can also quote QUIC [80] transport protocol. Even if this protocol is based on UDP, it has similarities with TCP, while tackling some of its weaknesses: reduced latency or better handling of packet losses. This makes QUIC a good candidate for LEO satellite transmissions;
- the scheduling policies can be tested with other kind of traffic or with mixtures of different traffics. Moreover, we can assess the performance of other scheduling policies and compare their performance to PF and CoDeS;
- the impact of short-lived flows, which represent actually the majority of the flows in the Internet, need to be considered more carefully, as the long RTT could be problematic for flows composed of only a dozen of packets. We also need to make sure that the mechanisms introduced in this thesis still work efficiently with these flows.

Appendix A

List of Publications

Published

- B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, L. Clarac, and N. Kuhn, "Making H-ARQ suitable for a mobile TCP receiver over LEO satellite constellations", in *International Conference on Wireless and Satellite Systems*, September 2017.
- B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, and N. Kuhn, "Impact of Delayed Acknowledgment on TCP performance over LEO satellite constellations", in *Fifth Federated and Fractionated Satellite Systems Workshop*, November 2017.
- B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, and N. Kuhn, "Controlled Delay Scheduler for VoIP over LEO constellations on LMS channels", in *Advanced Satellite Multimedia Systems Conference*, September 2018.

Under review

- B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, and N. Kuhn, "Impact of Delayed Acknowledgment on TCP performance over LEO satellite constellations", in *Journal of Spacecrafts and Rockets* (second round of reviews).

To be submitted

- Impact of scheduling policy on TCP performance (Chapter 6), in *International Journal of Satellite Communications and Networking*.

Bibliography

[1] F. P. Fontan, M. Vazquez-Castro, C. E. Cabado, J. P. Garcia, and E. Kubista, "Statistical modeling of the LMS channel," *IEEE Transactions on Vehicular Technology*, vol. 50, pp. 1549–1567, Nov 2001. (Cited in pages 3, 28, 33 et 34.)

[2] N. Blaunstein, Y. Cohen, and M. Hayakawa, "Prediction of fading phenomena in land-satellite communication links," *Radio Science*, vol. 45, no. 6, 2010. (Cited in pages 3, 28 et 33.)

[3] F. Perez-Fontan, M. A. Vazquez-Castro, S. Buonomo, J. P. Poiares-Baptista, and B. Arbesser-Rastburg, "S-band LMS propagation channel behaviour for different environments, degrees of shadowing and elevation angles," *Institute of Electrical and Electronics Engineers Transactions on Broadcasting*, vol. 44, pp. 40–76, Mar 1998. (Cited in pages 3, 4, 28, 33 et 59.)

[4] R. A. Ahmad, J. Lacan, F. Arnal, M. Gineste, and L. Clarac, "Enhanced HARQ for Delay Tolerant Services in Mobile Satellite Communications," in *The Seventh International Conference on Advances in Satellite and Space Communications SPACOMM 2015*, (Barcelona, ES), pp. pp. 1–6, April 2015. (Cited in pages 3 et 39.)

[5] X. Chen, Z. Fei, J. Kuang, and W. Sun, "Prediction of hybrid-ARQ based on mutual information model for LDPC coded OFDM system," in *2008 11th IEEE International Conference on Communication Technology*, pp. 700–703, Nov 2008. (Cited in page 3.)

[6] "*ns*-2.35." `https://www.isi.edu/nsnam/ns/doc/everything.html`, 2011. (Cited in pages 4 et 60.)

[7] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm." RFC 2582 (Experimental), Apr. 1999. Obsoleted by RFC 3782. (Cited in pages 5 et 50.)

[8] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-friendly High-speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 64–74, July 2008. (Cited in pages 5, 50 et 68.)

[9] R. A. Ahmad, *Mécanismes de fiabilité bi-directionnels couches basses pour les communications par satellite*. PhD thesis, 2016. Thèse de doctorat dirigée par

Lacan, Jérôme Réseaux, Télécom, Système et Architecture Toulouse, ISAE 2016. (Cited in page 6.)

[10] R. Braden, "Requirements for Internet Hosts - Communication Layers." RFC 1122 (INTERNET STANDARD), Oct. 1989. Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864. (Cited in pages 7, 44 et 74.)

[11] G. Judd, "Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter," in *Networked Systems Design and Implementation*, pp. 145–157, 2015. (Cited in pages 7, 45 et 74.)

[12] C. Caini and R. Firrincieli, "TCP Hybla: a TCP enhancement for heterogeneous networks," *International journal of satellite communications and networking*, vol. 22, no. 5, pp. 547–566, 2004. (Cited in pages 8, 52 et 79.)

[13] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, Mar 1998. (Cited in pages 13 et 91.)

[14] S. Shakkottai and A. L. Stolyar, "Scheduling algorithms for a mixture of real-time and non-real-time data in HDR," in *Teletraffic Engineering in the Internet Era* (J. M. de Souza, N. L. da Fonseca, and E. A. de Souza e Silva, eds.), vol. 4 of *Teletraffic Science and Engineering*, pp. 793 – 804, Elsevier, 2001. (Cited in pages 13 et 91.)

[15] S. Shakkottai and A. L. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *Translations of the American Mathematical Society-Series 2*, vol. 207, pp. 185–202, 2002. (Cited in pages 13 et 91.)

[16] G. Fairhurst, G. Giambene, S. Giannetti, C. P. Niebla, and A. Sali, "Cross-layer study for resource management in DVB-S2 with mobile users," in *2009 International Workshop on Satellite and Space Communications*, pp. 257–261, Sept 2009. (Cited in pages 13 et 91.)

[17] E. Chaput, M. Verloop, A. L. Beylot, and C. Baudoin, "Utility function based packet scheduling over DVB-S2," in *2013 IEEE International Conference on Communications (ICC)*, pp. 4288–4292, June 2013. (Cited in pages 13 et 91.)

[18] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 34, Aug. 2004. (Cited in pages 14 et 93.)

[19] ITU T-REC-G.114 Series G: Transmission systems and media, digital systems and networks, International telephone connections and circuits, General Recommendations on the transmission quality for an entire international telephone connection, one-way transmission time, May 2003. (Cited in pages 15, 35, 96 et 101.)

[20] B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, L. Clarac, and N. Kuhn, "Making H-ARQ suitable for a mobile TCP receiver over LEO satellite constellations," in *International Conference on Wireless and Satellite Systems*, pp. 33–42, Springer, 2017. (Cited in page 24.)

[21] B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, and N. Kuhn, "Impact of Delayed Acknowledgment on TCP performance over LEO satellite constellations," 2017. (Cited in page 25.)

[22] B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, and N. Kuhn, "Controlled Delay Scheduler for VoIP over LEO constellations on LMS channels," in *9th Advanced Satellite Multimedia Systems Conference*, 2018. (Cited in page 25.)

[23] J. Postel, "Transmission Control Protocol." RFC 793 (INTERNET STANDARD), Sept. 1981. Updated by RFCs 1122, 3168, 6093, 6528. (Cited in page 28.)

[24] Y. Chotikapong, H. Cruickshank, and Z. Sun, "Evaluation of TCP and Internet traffic via low Earth orbit satellites," *IEEE Personal Communications*, vol. 8, pp. 28–34, Jun 2001. (Cited in page 30.)

[25] L. Wood, "SaVi: satellite constellation visualization," in *First Annual CCSR Research Symposium - CRS 2011*, 2012. (Cited in pages 30 et 59.)

[26] "Iridium NEXT." `https://www.iridium.com/network/iridium-next/`, 2017. (Cited in pages 31 et 32.)

[27] "OneWeb." `http://www.oneweb.world/`, 2018. (Cited in pages 31 et 32.)

[28] "ORBCOMM." `https://www.orbcomm.com/en/networks/satellite/orbcomm-og2`, 2018. (Cited in page 31.)

[29] R. Suzuki and Y. Yasuda, "Study on ISL network structure in LEO satellite communication systems," *Acta Astronautica*, vol. 61, no. 7, pp. 648 – 658, 2007. (Cited in page 31.)

[30] ITU T-REC-G.1020, Performance parameter definitions for quality of speech and other voiceband applications utilizing IP networks. (Cited in pages 35, 36 et 99.)

[31] A. Hore and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369, Aug 2010. (Cited in page 36.)

[32] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2, pp. 1064–1070 vol.2, May 1993. (Cited in page 36.)

[33] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, January 1962. (Cited in page 36.)

[34] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, pp. 399–431, Mar 1999. (Cited in page 36.)

[35] M. Luby, "Lt codes," p. 271, IEEE, 2002. (Cited in page 36.)

[36] A. Shokrollahi, "Raptor codes," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 2551–2567, June 2006. (Cited in page 36.)

[37] F. Peng, Á. S. Cardona, K. Shafiee, and V. C. Leung, "TCP performance evaluation over GEO and LEO satellite links between Performance Enhancement Proxies," in *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, pp. 1–5, IEEE, 2012. (Cited in page 41.)

[38] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations." RFC 3135 (Informational), June 2001. (Cited in pages 41 et 52.)

[39] R. Wang, T. Taleb, A. Jamalipour, and B. Sun, "Protocols for reliable data transport in space internet," *IEEE Communications Surveys Tutorials*, vol. 11, pp. 21–32, Second 2009. (Cited in page 41.)

[40] N. Assaf, J. Luo, M. Dillinger, and L. Menendez, "Interworking between IP security and performance enhancing proxies for mobile networks," *IEEE Communications Magazine*, vol. 40, pp. 138–144, May 2002. (Cited in page 41.)

[41] V. Jacobson and R. Braden, "TCP extensions for long-delay paths." RFC 1072 (Historic), Oct. 1988. Obsoleted by RFCs 1323, 2018, 6247. (Cited in page 41.)

[42] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance." RFC 1323 (Proposed Standard), May 1992. Obsoleted by RFC 7323. (Cited in page 41.)

[43] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger, "TCP Extensions for High Performance." RFC 7323 (Proposed Standard), Sept. 2014. (Cited in page 42.)

[44] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 67–82, July 1997. (Cited in page 42.)

[45] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's Retransmission Timer." RFC 6298 (Proposed Standard), June 2011. (Cited in pages 42 et 83.)

[46] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options." RFC 2018 (Proposed Standard), Oct. 1996. (Cited in page 44.)

[47] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Conference on Computer Communications. Nineteenth Annual Joint Conference of the Institute of Electrical and Electronics Engineers Computer and Communications Societies*, vol. 3, pp. 1157–1165, 2000. (Cited in pages 45, 46 et 78.)

[48] M. Allman, "TCP Congestion Control with Appropriate Byte Counting (ABC)." RFC 3465 (Experimental), Feb. 2003. (Cited in page 45.)

[49] E. Altman, K. Avrachenkov, and C. Barakat, "TCP in presence of bursty losses," *Performance Evaluation*, vol. 42, no. 2, pp. 129 – 147, 2000. (Cited in pages 46 et 79.)

[50] R. Sallantin, C. Baudoin, E. Chaput, F. Arnal, E. Dubois, and A. L. Beylot, "Initial spreading: A fast Start-Up TCP mechanism," in *38th Annual IEEE Conference on Local Computer Networks*, pp. 492–499, Oct 2013. (Cited in pages 46 et 78.)

[51] D. Ciullo, M. Mellia, and M. Meo, "Two schemes to reduce latency in short lived TCP flows," *IEEE Communications Letters*, vol. 13, pp. 806–808, October 2009. (Cited in page 46.)

[52] Y. Cheng, N. Cardwell, N. Dukkipati, and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP." `https://tools.ietf.org/html/draft-ietf-tcpm-rack-03`, 2018. (Cited in page 46.)

[53] N. Dukkipati, N. Cardwell, Y. Cheng, and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses." `https://tools.ietf.org/html/draft-dukkipati-tcpm-tcp-loss-probe-01`, 2013. (Cited in page 48.)

[54] I. Bisio, M. Marchese, and M. Mongelli, "Performance Enhanced Proxy Solutions for Satellite Networks: State of the Art, Protocol Stack and Possible Interfaces," in *Personal Satellite Services* (K. Sithamparanathan and M. Marchese, eds.), (Berlin, Heidelberg), pp. 61–67, Springer Berlin Heidelberg, 2009. (Cited in page 51.)

[55] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, (New York, NY, USA), pp. 287–297, ACM, 2001. (Cited in page 51.)

[56] L. A. Grieco and S. Mascolo, "Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 25–38, Apr. 2004. (Cited in page 51.)

[57] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 307–321, June 2001. (Cited in page 52.)

[58] C. Roseti and E. Kristiansen, "TCP Noordwijk: TCP-based transport optimized for web traffic in satellite networks," in *26th International Communications Satellite Systems Conference (ICSSC)*, 2008. (Cited in page 53.)

[59] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, September-October, pp. 20 – 53, 2016. (Cited in page 54.)

[60] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue*, vol. 9, pp. 40:40–40:54, Nov. 2011. (Cited in page 54.)

[61] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP." RFC 3168 (Proposed Standard), Sept. 2001. Updated by RFCs 4301, 6040. (Cited in page 54.)

[62] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet." RFC 2309 (Informational), Apr. 1998. Obsoleted by RFC 7567, updated by RFC 7141. (Cited in page 54.)

[63] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger, "Enabling Internet-Wide Deployment of Explicit Congestion Notification," in *Passive and Active Measurement* (J. Mirkovic and Y. Liu, eds.), (Cham), pp. 193–205, Springer International Publishing, 2015. (Cited in page 54.)

[64] B. Briscoe, R. Woundy, and A. Cooper, "Congestion Exposure (ConEx) Concepts and Use Cases." RFC 6789 (Informational), Dec. 2012. (Cited in page 54.)

[65] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP." RFC 2883 (Proposed Standard), July 2000. (Cited in page 55.)

[66] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control." RFC 2581 (Proposed Standard), Apr. 1999. Obsoleted by RFC 5681, updated by RFC 3390. (Cited in pages 55 et 63.)

[67] "*ns*-3." `https://www.nsnam.org/`, 2018. (Cited in page 60.)

[68] "*sns*-3." `http://satellite-ns3.com/`, 2018. (Cited in page 60.)

[69] D. X. Wei and P. Cao, "NS-2 TCP-Linux: An NS-2 TCP Implementation with Congestion Control Algorithms from Linux," in *Proceeding from the 2006 Workshop on Ns-2: The IP Network Simulator*, WNS2 '06, (New York, NY, USA), ACM, 2006. (Cited in page 62.)

[70] L. Schulte, A. Zimmermann, P. Nanjundaswamy, L. Eggert, and J. Manner, "I'll be a bit late - packet reordering in mobile networks," *Journal of Communications and Networks*, vol. 19, pp. 693–703, Dec 2017. (Cited in page 71.)

[71] J. Chen, M. Gerla, Y. Z. Lee, and M. Sanadidi, "TCP with delayed ack for wireless networks," *Ad Hoc Networks*, vol. 6, no. 7, pp. 1098 – 1116, 2008. (Cited in page 74.)

[72] L. Wood, G. Pavlou, and B. Evans, "Effects on TCP of routing strategies in satellite constellations," *Institute of Electrical and Electronics Engineers Communications Magazine*, vol. 39, pp. 172–181, Mar 2001. (Cited in pages 74 et 77.)

[73] J. B. Dupé, E. Chaput, C. Baudoin, C. Bès, A. Deramecourt, and A. L. Beylot, "Rule-based packet scheduling for DVB-S2 through generic stream encapsulation," in *2014 IEEE International Conference on Communications (ICC)*, pp. 3576–3581, June 2014. (Cited in page 91.)

[74] J. B. Dupé, E. Chaput, C. Baudoin, C. Bès, A. Deramecourt, and A. L. Beylot, "Optimized GSE packet scheduling over DVB-S2," in *2014 IEEE Global Communications Conference*, pp. 2856–2861, Dec 2014. (Cited in page 91.)

[75] G. Ash and D. McDysan, "Generic Connection Admission Control (GCAC) Algorithm Specification for IP/MPLS Networks." RFC 6601 (Experimental), Apr. 2012. (Cited in page 95.)

[76] G. Sarwar, R. Boreli, and E. Lochin, "On the quality of VoIP with DCCP for satellite communications," *International Journal of Satellite Communications and Networking*, vol. vol. 30, pp. pp. 163–180, 2012. (Cited in page 95.)

[77] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A Quantitative Measure of Fairness and Discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984. (Cited in page 114.)

[78] Z. Ding, Y. Liu, J. Choi, Q. Sun, M. Elkashlan, C. I, and H. V. Poor, "Application of Non-Orthogonal Multiple Access in LTE and 5G Networks," *IEEE Communications Magazine*, vol. 55, pp. 185–191, February 2017. (Cited in page 116.)

[79] F. Liu, P. Mahonen, and M. Petrova, "Proportional fairness-based user pairing and power allocation for non-orthogonal multiple access," in *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1127–1131, Aug 2015. (Cited in page 116.)

[80] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport." `https://tools.ietf.org/html/draft-ietf-quic-transport-14`, 2018. (Cited in page 118.)

Résumé : L'accès à Internet par satellite permet de connecter des régions isolées de la terre ou des utilisateurs en mouvement, pour lesquels une solution terrestre peut s'avérer coûteuse voire impossible à déployer. L'utilisation de constellations de satellites en orbite basse (LEO) permet de plus de transmettre avec des délais similaires à ceux des transmissions terrestres, permettant l'utilisation de la pile TCP/IP standard. L'utilisation d'un tel environnement engendre cependant des contraintes spécifiques à ce type de réseau et bandes de fréquences utilisées, comme un important taux d'erreur paquet, à cause de la traversée l'atmosphère, de la mobilité de récepteur ou d'interférences. Pour compenser ces forts taux de pertes, des mécanismes de fiabilisation doivent être introduits au niveau des liens les plus difficiles. Ces mécanismes ont toutefois un impact négatif sur les performances des protocoles de transport, notamment TCP, limitant grandement le débit. Le but de cette thèse est d'effectuer tout d'abord une étude de performance de TCP, et de proposer des solutions afin d'améliorer la performance des transmissions dans notre scénario. Dans un deuxième temps, nous verrons comment déployer efficacement différents services internet via des constellations LEO de satellites de façon efficace.

Abstract : LEO satellite constellations allow to connect isolated or mobile users to the Internet, when terrestrial solutions are too expensive or impossible to deploy. Using such constellations allow to connect these areas with transmission delays close to terrestrial delays, and then to use the standard TCP/IP stack. However, this environment brings new impairments such as an important error rate between the satellites and the ground receivers. To counteract this high error rate, reliability schemes need to be introduced on this link. However, these schemes have a negative impact on the transport protocol (TCP) performance, mitigating the transmission throughput. In this thesis, we first perform a performance analysis of TCP in our scenario, and propose some solutions to improve transmission performance. Then we study solutions to efficiently deploy several Internet services over LEO constellations.

Keywords : TCP, Reliability schemes, HARQ, SATCOM, Mobile services, LMS