# Contrôle de transmission dans les réseaux satellites

Nesrine Chaher

3 juillet 1998

# Table des matières

# Table des figures

# Première partie

# Présentation du sujet

Le stage a pour objectif d'analyser et d'optimiser les mécanismes de correction d'erreurs et de contrôle de flux (TCP/IP) pour des réseaux satellites.

Le protocole TCP permet d'assurer un contrôle de transmission fiable de données de bout en bout indépendamment des spécificités des différentes liaisons. Cependant les caractéristiques des canaux utilisant des satellites géostationnaires telles qu'un délai élevé, un produit délai par bande passante élevé et un BER[1] élevé de l'ordre de $10^{-7}$ font que TCP n'atteint pas les performances terrestres car à l'origine TCP a été conçu pour des réseaux pour lesquels la taille des buffers est bien plus grande que le produit délai par bande passante. On voudrait améliorer le comportement de TCP dans ces environnements.

1. Bit Error Rate

# Deuxième partie

# TCP sur les liens satellites

La première partie du stage consistait en l'étude du protocole TCP et l'identification des problèmes dûs à l'impact des caractéristiques des liens satellites sur le protocole ainsi que les différents mécanismes de correction d'erreurs.

# Chapitre 1

# TCP

Les algorithmes utilisés par TCP[Jac] pour le contrôle de congestion sont slow start, congestion avoidance, fast retransmit et fast recovery. Leur but étant de contrôler la quantité de données non acquittées en cours de transmission et de retransmettre les données perdues. Les vieilles versions de TCP utilisaient un algorithme différent pour injecter les données dans le réseau, il consistait simplement à envoyer autant de données que la fenêtre du récepteur le permettait. Ceci peut s'avérer suffisant si les deux hôtes se situent dans le même LAN[1] mais est néfaste s'il y a des routeurs ou des liens lents entre les hôtes. Car les routeurs intermédiaires doivent mémoriser les paquets, et il est possible que ce routeur n'ait pas assez de mémoire.

En utilisant l'algorithme de slow start, la fenêtre initiale de l'émetteur est initialisée à un, TCP doit attendre le retour de l'acquittement avant de continuer à transmettre et sa fenêtre sera augmentée d'un segment pour chaque acquittement reçu et ce jusqu'à ce que l'on atteigne le seuil ssthresh[2]. Là on passe à la phase de congestion avoidance, phase pendant laquelle la fenêtre sera augmentée d'un segment environ par RTT[3]. Si une perte de paquet se produit, détectée par l'expiration d'un timer ou la réception de trois ACKs dupliqués, on réduit la taille de la fenêtre. Une congestion peut se produire quand plusieurs flux de données arrivent à un routeur qui a une capacité inférieure à la somme de ces flux entrants ou quand les données arrivent sur un grand canal (LAN rapide) pour ressortir sur un plus petit (WAN[4] lent).

TCP détecte la congestion de deux manières différentes : soit par l'expiration d'un timer ou par la recéption de trois acquittements dupliqués. Dans le premier cas, TCP n'a aucune connaissance de l'état du réseau et doit procéder à l'envoi des données d'une manière conservative en utilisant slow start. Alors qu'en cas de retransmission due à la réception de trois acquittements dupliqués, TCP sait qu'il y a un flux de paquets circulant dans le réseau et peut en déduire que la congestion ne doit pas être si grave. La raison pour laquelle on attend de recevoir trois acquittements dupliqués avant de déduire une perte de paquet est qu'en recevant un nombre inférieur à ce seuil fixé à trois, il peut s'agir uniquement

1. Local Area Network.
2. Le seuil ssthresh est la valeur à partir de laquelle on entre dans la phase de congestion avoidance.
3. Round Trip Time est le delai aller/retour d'un paquet.
4. Wide area Network.

d'un réarrangement de paquets et donc pas de paquet perdu.

L'algorithme du Fast Recovery est utilisé entre le moment où un segment est retransmis en utilisant le Fast Retransmit jusqu'à ce que tous les paquets perdus soient retransmis, ce mécanisme est bien mieux que TCP seul mais moins bien que TCP avec les SACKs. L'avantage de ce mécanisme comparé aux SACKS est qu'il n'a besoin d'être implémenté que chez l'émetteur. Le mécanisme des SACKs [5] permet au récepteur d'informer l'émetteur des paquets reçus; cela diminue le nombre de paquets retransmis.

FACKS est un mécanisme qui utilise les SACKS, et qui remplace le Fast Recovery. Il permet de réduire le débit graduellement en cas de pertes de paquets. TCP standard ne le permet pas, car l'émetteur ne sait pas quels segments retransmettre. FACK utilise le mécanisme "rate halving" pour injecter de nouveaux paquets ou des paquets retransmis dans le réseau. La quantité de données envoyées détermine la fenêtre de congestion après la phase "Fast Recovery".

Les différentes versions de TCP:

- TCP Tahoe est la version standard.
- TCP Reno est une version qui a eté standardisée dans [Ste97], qui est la plus répandue dans les systèmes UNIX. Elle comprend Fast Retransmit et Fast Recovery, Slow start et congestion avoidance. L'émetteur TCP entre dans la phase Fast Recovery après avoir reçu trois acquittements dupliqués, il retransmet le (premier) paquet perdu et réduit sa fenêtre de congestion de moitié; au lieu d'entrer dans la phase de slow start (Tahoe), l'émetteur Reno utilise les acquittements reçus pour régler l'émission des paquets. Durant la phase Fast recovery l'émetteur augmente sa fenêtre de congestion du nombre d'ACKs dupliqués reçus (en notant que chaque acquittement dupliqué reçu représente un paquet qui a quitté le réseau et qui maintenant est chez le récepteur). Après être entré dans la phase Fast Recovery et retransmis un paquet, l'émetteur commence à envoyer un paquet pour chaque acquittement dupliqué reçu. A la réception d'un "recovery ACK", un acquittement pour de nouvelles données, l'émetteur quitte la phase de Fast Recovery. On remet la fenêtre de congestion à ssthresh.
- New Reno est une extension de Reno qui quitte la phase de retransmission quand le segment (non acquitté) ayant le numéro de séquence le plus élevé est enfin acquitté. Il inclut un petit changement à l'émetteur qui élimine l'attente de l'expiration d'un timer si plusieurs paquets sont perdus. Le changement concerne le comportement de l'émetteur durant Fast Recovery quand un ACK partiel [6] est reçu. Dans Reno, la réception d'un ACK partiel fait sortir l'émetteur de la phase Fast Recovery en remettant la fenêtre utilisable à la valeur de la fenêtre de congestion (ssthresh). Dans New Reno, les ACKs partiels sont vus comme une indication de perte du paquet suivant immédiatement le paquet acquitté et qui doit être retransmis. Si plusieurs paquets sont perdus (de la même fenêtre), l'émetteur n'est pas tenu d'attendre l'expiration d'un timer avant de retransmettre, en retransmettant un paquet par RTT. New Reno reste dans la phase Fast Recovery jusqu'à ce que tous les paquets soient acquittés.

5. Selective ACKnowledgment.
6. Un ACK partiel est un ACK qui acquitte quelques paquets.

Comparé à TCP Tahoe, l'algorithme de Fast Recovery de TCP Reno est optimal si un paquet est perdu par fenêtre de données. L'émetteur Reno retransmet au maximum un paquet par RTT. En revanche, il subit une détérioration de performance si plusieurs paquets appartenant à la même fenêtre sont perdus.

## 1.1 Caractéristiques des liens satellites

La particularité du lien satellite est le délai élevé, les satellites géostationnaires sont à une altitude de 36000 km et ont une période de rotation identique à celle de la terre; ce qui permet aux stations terrestres de toujours voir le satellite à la même position, de plus un géostationnaire a une couverture constante, il n'y a pas de Handover [7].

Parmi les avantages des satellites (comparés à la fibre optique), il y a le broadcast naturel qui favorise leur utilisation pour les applications de type multicast, ainsi que le pouvoir de joindre des régions éloignées ou des utilisateurs mobiles.

Les caractéristiques les plus importantes sont:

- le délai de propagation qui est compris entre 239.6 ms et 279 ms. Ce délai correspond au délai d'un hop (saut), le RTT est donc de 558ms environ. Parmi les paramètres affectant le RTT, il y a le temps de transmission sur les autres liens dans le réseau ainsi que le délai d'attente dans les buffers le long de la connexion.
- le bruit : la puissance d'un signal radio décroit proportionnellement au carré de la distance parcourue, du fait de l'éloignement important des satellites GEO, le signal s'affaiblira avant d'atteindre la destination, sans compter que les fréquences sont sujettes aux conditions atmosphériques comme la pluie, il en résulte des taux d'erreurs élevés de l'ordre de $10^{-7}$.
- la bande passante : c'est une ressource limitée et les allocations de fréquences ou de bande passante se font par des accords internationaux.

## 1.2 Impact des caractéristiques des liens satellites sur le protocole TCP

Les caratéristiques physiques du lien satellite dégradent les performances de TCP :

- délai: le délai élevé fait que l'émetteur met beaucoup de temps avant de déterminer si son paquet est arrivé ou non; ce délai affecte les applications interactives comme le "telnet".
- le produit délai par bande passante: il détermine la quantité d'informations que le protocole doit avoir "en vol" (des données envoyées mais non encore

7. Dans le cas de constellations de satellites qui comprennent des satellites LEO : Low Earth Orbit ou MEO : Medium Earth Orbit, où la zone couverte n'est pas constante il faut qu'il y ait un relai de satellites.

acquittées) pour utiliser au maximum le lien. Avec un lien satellite, TCP devra maintenir une quantité importante "en vol".

- les erreurs de transmission: TCP va interpréter toute perte de paquet comme un signal de congestion et va réduire sa fenêtre de transmission (donc son débit) même s'il n y a pas de congestion.
- l'utilisation asymétrique: pour des raisons de coûts dûs à la taille de l'antenne ou à la puissance de transmission, le lien montant a une capacité moins importante que le descendant; cette asymétrie affecte le comportement de TCP.

Le gros problème des liens satellites provient des erreurs de transmissions. Pour toutes ces raisons, il s'agit d'optimiser TCP.

## 1.3 Comment améliorer TCP?

Des mécanismes extérieurs au protocole TCP ont été proposés pour améliorer le comportement de TCP.

### 1.3.1 Le Path MTU Discovery

Cette technique permet de déterminer la taille de paquet maximale que l'on peut utiliser durant une connexion TCP sans que ce paquet ne soit fragmenté (par la couche IP). Eviter la fragmentation est important car cela induirait beaucoup d' "overhead" (surcoût) dû aux en-têtes.

L' inconvénient est que cela peut prendre beaucoup de temps avant de trouver la taille maximale (si le chemin contient plusieurs liens), temps durant lequel TCP n'envoie pas de données (très pénalisant pour les transferts courts). En pratique, le gain est quand même important grâce à l'utilisation de valeurs standards de MTU[8]. Si on utilise de plus grands segments dans un environnement où le BER est élevé, la probabilité que le segment arrive erroné, augmente avec la taille du segment, donc l'utilisation de plus petits segments peut assurer que plus de segments seront transmis et moins de données seront retransmises. Il y a bien une relation entre le BER et la taille du segment qui varie selon les caractéristiques du canal de transmission.

### 1.3.2 Forward Error Correction (FEC)

L'idéal serait de pouvoir différencier les erreurs de congestion de celles de transmission. Plusieurs recherches sont en cours pour utiliser des notifications de congestion explicites (ECN[9]) dans le but d'empêcher TCP de diminuer sa fenêtre sans qu'il y ait de congestion et de retransmettre les paquets corrompus sans diminuer le débit. Une autre solution est de rendre le lien aussi sûr que possible c'est à dire qu'il faut utiliser le codage FEC [nA93] et [Riz97] pour améliorer le taux d'erreur du lien satellite.

8. Maximum Transmission Unit.
9. Explicit Congestion Notification.

L'inconvénient de la FEC est qu'elle n'est pas sans coût car elle consomme une partie de la bande passante (pour la redondance) et peut augmenter la gigue[10] à cause des temps de traitement dûs au codage/décodage.

## 1.4 Les modifications proposées dans la littérature à l'algorithme de slow start

Ces modifications ont pour but de réduire la quantité de bande passante perdue pour améliorer le comportement de TCP.

### 1.4.1 Améliorer TCP durant slow start

Si on s'intéresse à la phase slow start, on doit avoir assez d'espace dans les buffers intermédiaires.

Le protocole TCP va mettre beaucoup de temps avant d'atteindre le débit maximum et une des causes possibles de ce délai est le manque d'espace buffer. Durant la phase de slow start, pour chaque acquittement reçu, deux segments sont envoyés. L'émetteur transmet à une vitesse qui est le double de celle des segments acquittés. Si les liens sont sûrs, la perte du premier paquet est dûe au dépassement de la capacité du buffer. On voudrait que cette perte se produise dans le RTT suivant celui où la fenêtre de congestion a atteint le produit délai par bande passante. Pour éviter de perdre des paquets, le routeur doit avoir une taille de buffer égale à la moitié du produit délai par bande passante. Dans les réseaux à hauts débits, cette condition ne pouvant être satisfaite, il en résulte que TCP va sortir de la phase slow start prématurément c'est à dire quand la taille du buffer sera dépassée. Les conséquences sont assez sévères car TCP va ensuite entrer dans la phase congestion avoidance, phase pendant laquelle la fenêtre de congestion va augmenter plus lentement. Donc, même si le canal est libre on transmettra à un débit faible jusqu'à ce que l'algorithme de congestion avoidance nous permette d'envoyer assez de données pour remplir le lien.

Dans un réseau avec un lien satellite, après l'envoi du premier paquet, l'émetteur doit attendre une période de 500ms avant de pouvoir continuer, période durant laquelle on attend. Cette attente encourage l'utilisation du Path MTU Discovery car la quantité de données qu'on met dans le paquet n'est pas déterminée par TCP qui ne contrôle que le nombre de segments envoyés. Donc utiliser de plus gros paquets permet à TCP d'envoyer plus de données et on obtient une meilleure utilisation de la bande passante.

Les modifications suivantes ont été acceptées par l'IETF :[11] [MA98b]

### 1.4.2 Utilisation de larges fenêtres

On sait que le débit de TCP dans un environnement avec un lien satellite est limité par la formule suivante:

$$\text{débit} = \frac{\text{taille de la fenêtre}}{\text{RTT}}. \tag{1.1}$$

10. La gigue est le retard introduit par le décodage.
11. Internet Engineering Task Force.

Le débit se limite donc à 117.027 bytes/s. La taille de la fenêtre est limitée à 64KB, les valeurs les plus courantes varient de 4KB à 24KB.

Une seule connexion TCP ne peut pas utiliser le lien au maximum. Exemple de liens : le lien T1 qui a une bande passante de 192000 bytes/s. Pour pallier à la limitation due au fait que la taille de la fenêtre est codée sur 16 bits et donc ne peut dépasser 64KB, l'IETF [MA98a] a proposé une option qui permet de multiplier la taille de la fenêtre par une constante. Le multipliant a une valeur maximum de $2^{14}$ octets car la taille de la fenêtre est également limitée par l'espace des numéros de séquences : la fenêtre ne doit pas dépasser la moitié de cet espace pour pouvoir sans ambiguité savoir si un octet est à l'intérieur ou à l'extérieur de cet espace. La taille maximum de la fenêtre est donc de $2^{30}$ octets et le débit maximum de 15Gbits/s. Sachant que l'on atteint des débits en Terabits/s, cette valeur est petite et en l'absence de modification à l'en-tête TCP, on ne pourra pas faire mieux.

### 1.4.3 Utilisation des acquittements sélectifs (SACKs)

Ceci permettrait d'éviter les retransmissions inutiles en donnant la possibilité au récepteur de communiquer à l'émetteur les paquets reçus, de plus on se remet des paquets perdus sans diminuer le débit (on ne diminue pas la taille de la fenêtre). En l'absence des SACKs qui ont été proposés dans [MMR96], l'émetteur doit déduire les paquets perdus et cela induit des retransmissions inutiles. De plus, si on utilise de larges fenêtres, on aura plus de paquets perdus par fenêtre et dans ce cas l'utilisation des SACKs permettrait à l'émetteur de ne retransmettre que les paquets perdus plus rapidement (car on n'attend pas l'expiration d'un timer). On ne peut retransmettre qu'un paquet perdu par fenêtre avec TCP Tahoe, si plusieurs pertes de paquets se produisent dans une même fenêtre, l'émetteur sera obligé d'attendre l'expiration d'un timer avant de pouvoir déterminer les paquets perdus.

Les modifications suivantes ont été proposées par le groupe de recherche "end-to-end" de l'IETF mais pas encore adoptées :

### 1.4.4 Augmenter la taille de la fenêtre initiale de TCP

La proposition est d'augmenter la taille de la fenêtre initiale [MA98a] pour qu'elle passe d'un segment à 2 ou 4 segments, la taille proposée est donnée par :

$$\min\{4 * \mathrm{MSS}, \max\,(2 * \mathrm{MSS}, 4380 \text{ octets})\}. \tag{1.2}$$

Cette modification ne s'appliquerait qu'au début du transfert et pas après une phase de slow start declenchée par la perte d'un paquet qui serait détectée par l'expiration d'un timer.
Les avantages de cette modification:

– en terme de temps de transfert:

La durée de slow start est déterminée par la formule suivante :

$$\text{temps de slow start} = Rlog_2 W_A. \tag{1.3}$$

$R$ étant le round trip time (délai aller/retour).
$W_A$ la taille de fenêtre maximale (donnée par le récepteur). Avec une fenêtre initiale de $W_I$, cette durée sera réduite à

$$\text{temps de slow start } = R(logW_A - logW_I). \tag{1.4}$$

Le temps requis pour atteindre une fenêtre de taille $W_A$ est diminué d'un nombre de RTT égal au nombre nécessaire pour atteindre $W_I$.

Cette modification réduit le temps de transfert pour les applications avec peu de données à transmettre, et surtout sur des liens à produit délai par bande passante élevé, cette modification élimine jusqu'à trois RTT et, si le récepteur implémente le mécanisme des delayed ACKS, un delayed timeout durant la phase initiale de slow start. Le bénéfice pour les transferts plus longs (bulk) est moins significatif car comme la longueur du transfert augmente, l'impact de slow start sur la performance totale diminue.

– en terme de bande passante: réduit la quantité de bande passante perdue et réduit le temps que va passer TCP dans la phase slow start.

Les inconvénients pour une seule connexion:

– si on se trouve dans un environnement congestionné, on risque d'aggraver la congestion; il y a quelques années un des liens trans-atlantiques était tellement surchargé que la taille de fenêtre qui convenait était d'un segment. Dans cet environnement, une fenêtre initiale de quatre segments aggrave la congestion. Il y a des situations où une connexion TCP commençant avec une fenêtre initiale d'un segment ne perd pas de paquets alors qu'en commençant avec 4 segments elle peut engendrer des pertes de paquets.

– si le routeur est incapable de traiter de courts bursts, on risque de passer prématurement à la phase de congestion avoidance et dans ce cas on n'utilisera jamais le lien à sa capacité maximum.

Les inconvénients pour un réseau:

– les larges fenêtres peuvent occasionner des pertes de paquets de la fenêtre initiale, donc des retransmissions inutiles, ce qui ne risque pas de se produire en utilisant le slow start traditionnel avec une fenêtre initiale d'un segment. Si on considère un réseau avec plusieurs liens congestionnés, on va retransmettre des segments sur un lien qui vont être perdus sur le lien suivant congestionné: on gaspille de la bande passante. De plus si le taux de perte de paquets est élevé, avec une fenêtre initiale plus grande on augmentera d'avantage ce taux, cela est dû aux buffers qui appliquent une politique FIFO et qui ne peuvent pas venir à bout des bursts. Mais dans la pratique et dans les réseaux à haut débit où 4K est plutôt une petite quantité, cette modification devrait être bénéfique, et en utilisant les buffers à technique RED[12], on pourra être plus tolérant aux bursts de trafic. Les risques de congestion vont plus être dûs au trafic UDP qui n'a aucun contrôle de congestion.

12. Random Early Detection: Les buffers implémentant ce mécanisme permettent aux routeurs intermédiaires d'informer l'émetteur TCP qu'une congestion se forme avant que cet émetteur n'envoie une quantité inappropriée d'informations.

Des tests ont été effectués [MA98c] pour étudier l'impact d'une fenêtre initiale plus grande sur une connexion mais pas sur les autres connexions TCP, cette omission est due en partie au fait que l'effet est négligeable car dans les réseaux actuels on subit des bursts de trafic dûs au mécanisme de delayed ACKS (un ACK qui peut couvrir jusqu'à deux paquets non acquittés). Mais on sait que dans des réseaux lents à buffers de petite taille l'effet est négatif. Les résultats des expériences montrent qu'on réduit le temps de transfert de près de 25% avec un taux de pertes qui n'augmente que de .04%.

### 1.4.5 Modifier l'algorithme de l'émetteur

Le mécanisme des delayed ACKs permet au récepteur de diminuer la vitesse d'envoi des acquittements en étant tenu d'en envoyer au moins un pour deux segments reçus et si le second segment n'est pas reçu au bout d'un délai (qui ne doit pas dépasser 500ms) un acquittement est généré. Dans un réseau satellite l'utilisation des delayed ACKs augmente le temps de transfert. Diminuer la fréquence d'envoi des acquittements a un impact négatif sur l'algorithme étant donné qu'il diminue le taux d'accroissement de la fenêtre et augmente le temps que va mettre slow start pour atteindre le seuil de ssthresh, cela est dû au fait que TCP se base sur le nombre d'ACKs reçus pour augmenter la taille de la fenêtre de congestion au lieu de se baser sur le nombre de segments acquittés.

La modification proposée [MAT98] est d'augmenter la fenêtre de congestion d'une valeur égale au nombre de segments non acquittés couverts par l'ACK reçu, au lieu de l'augmenter d'un pour chaque ACK reçu, la fenêtre ne sera jamais augmentée de plus de deux segments à la fois (pour ne pas violer RFC 1122). Cette modification serait utilisée durant la phase de slow start initiale ainsi que durant les phases de slow start qui suivent une expiration d'un timer due à une perte de segment.

### 1.4.6 Espacement des acquittements

Une autre proposition est l'espacement des acquittements [Par97] qui peut être effectué par le récepteur qui ralentit l'envoi des acquittements ou par le réseau i.e. les routeurs dans le but d'étaler les bursts. Si les routeurs étaient capables d'espacer les acquittements cela permettrait à l'émetteur de transmettre à la bonne vitesse sans modifications à apporter au protocole. Le problème est qu'il faut trouver l'algorithme qui détermine l'espacement adéquat qui va dépendre de l'émetteur, la valeur de l'espacement va varier selon qu'il se trouve dans la phase de slow start ou de congestion avoidance. Si comme c'est souvent le cas dans les réseaux satellites, les acquittements n'empruntent pas le même chemin que les segments envoyés par l'émetteur, l'espacement ne pourra pas être utilisé. Même si cela viole la philosophie d'Internet, l'idée est agréable car cela ne consisterait qu'en un changement dans les algorithmes des noeuds du réseau.

### 1.4.7 Estimer le seuil de threshold

Quand devons-nous arrêter la phase de slow start et passer à la phase de congestion avoidance? L'idéal serait d'arrêter la phase de slow start juste avant d'avoir une perte de paquet, et si on pouvait estimer la taille de la bande passante disponible on pourrait calculer la taille de fenêtre nécessaire qui serait:

$$\text{taille de fenêtre} = (\text{ bande passante }) * \text{RTT}. \quad (1.5)$$

Le seuil de ssthresh (qui détermine quand sortir de la phase de slow start) serait égal à cette valeur. Estimer le RTT est déjà fait mais pour ce qui est de l'estimation de la bande passante, c'est plus compliqué car il faut prendre en compte tous les utilisateurs du réseau à un moment donné et déterminer leur utilisation en terme de bande passante afin de déduire celle disponible. Hoe a proposé l'utilisation de l'algorithme de "packet pair" pour déterminer la valeur du ssthresh. Cet algorithme consiste en l'observation de l'espacement entre les premiers acquittements pour déterminer la bande passante du goulot d'étranglement. Ce mécanisme fonctionne dans les réseaux à liens symétriques, mais dans les réseaux à liens asymétriques le taux d'arrivée des acquittements n'est pas forcément égal à la bande passante dans l'autre direction. En général l'algorithme du "packet pair" est difficile à appliquer durant la phase de slow start car étant donné que le récepteur peut retarder l'envoi des acquittements, deux paquets envoyés l'un après l'autre ne vont pas forcément générer chez le récepteur l'envoi de deux acquittements immédiats. L'espacement des acquittements peut en effet induire une sous estimation de la bande passante. Hoe suggère de mesurer la bande passante en utilisant trois acquittements proches. Comme un récepteur TCP envoie un acquittement pour tous les deux segments, trois acquittements proches ne seront générés que si l'émetteur envoie six segments proches. Mais durant slow start l'émetteur ne peut envoyer six segments que s'il a reçu trois acquittements. La seule solution serait de commencer avec une fenêtre initiale de six segments; ce qui est inapproprié.

### 1.4.8 Compression des en-têtes TCP/IP

Plusieurs champs des en-têtes IP et TCP demeurent constants durant une session TCP, changent peu ou peuvent être déduits d'autres sources; par exemple l'adresse source et destination ainsi que la version du protocole sont des champs constants. La longueur du paquet peut être déduite des couches protocolaires plus basses. La compression des en-têtes a pour but de réduire l'overhead d'une session TCP en remplaçant ces champs par un numéro de connexion. En utilisant ce mécanisme, l'émetteur envoie un premier en-tête en incluant un numéro de connexion que l'émetteur utilisera ensuite pour référencer la connexion. Cette compression peut réduire la taille d'un en-tête TCP de 20 octets à 3 ou 4 octets. Cette réduction est surtout utile quand la ressource "produit par bande passante" est limitée comme dans les réseaux sans fils ou liens satellites mobiles. On peut encore gagner en codant les champs qui changent peu en envoyant la différence avec le précédent paquet. Le récepteur doit garder une copie de chaque en-tête reconstruit pour y appliquer la différence. Le problème de cet encodage est que si un paquet est perdu les paquets suivants ne pourront être décodés.

## 1.5 Solution niveau applicatif

Une solution niveau applicatif a été proposée pour résoudre le problème d'inefficacité de TCP sur les liens satellites [All97], elle consiste en XFTP qui utilise plusieurs connexions TCP, ainsi qu'un algorithme de congestion avoidance différent. L'avantage d'utiliser plusieurs connexions est que cela peut être implementé au niveau de l'application sans changement à effectuer au protocole TCP. XFTP a été developpé car les mécanismes récents proposés ne sont pas encore implementés, ni bien étudiés sur les liens satellites, ie: il n'y a pas encore d'implémentation TCP avec de larges fenêtres et la modification du mécanisme des acquittements ne sera pas disponible avant plusieurs années. Avec XFTP, il a été possible de simuler une fenêtre plus grande et l'algorithme de congestion avoidance a été modifié; l'utilisateur choisit une valeur initiale pour le nombre maximum de connexions pour chaque transfert de fichier, l'émetteur utilise des datagrammes UDP pour déterminer le RTT courant. En utilisant ces échantillons de RTT, XFTP va augmenter ou diminuer le nombre de connexions TCP. Les résultats de tests effectués montrent que sur des liens sans erreur, on obtient les mêmes performances (le même débit) qu'en utilisant une large fenêtre ou plusieurs connexions TCP, on voit que le nombre de connexions affecte le débit. Les avantages de XFTP:

- slow start est plus aggressif car XFTP multiplie la fenêtre de congestion par le nombre de connexions courantes ce qui conduit à une utilisation maximale de la capacité disponible, pour obtenir un résultat équivalent avec une connexion TCP il suffit d'incrémenter la fenêtre initiale de slow start qui est actuellement d'un segment.

- si l'option d'acquittements séléctifs est positionnée (SACKs), l'émetteur pourra détecter jusqu'à $N$ erreurs simultanément ($N$ étant le nombre de connexions TCP) et pourra les corriger avec Fast Retransmit.

- congestion avoidance est plus aggressif. On sait que dû au long RTT la fenêtre de congestion augmente lentement durant dette phase, mais avec XFTP la fenêtre augmentera plus vite (la vitesse sera multipliée par le nombre de connexions).

Les recherches en cours essayent de modifier l'algorithme de congestion avoidance pour prendre en compte les variations de RTT avant d'augmenter la fenêtre de congestion.

Mais le principal inconvénient de XFTP est le suivant : comment choisir le bon nombre de connexions car si on en utilise trop on va surcharger le réseau et subir beaucoup de pertes ?

# Chapitre 2

# Nouvelles améliorations de TCP pour le contrôle de congestion

Le rôle de TCP est de réguler le flux de paquets dans un réseau pour qu'il n'y ait pas de congestion et pour que les ressources soient équitablement partagées entre les différentes sources. Cette régulation se fait par la modification dynamique de la taille de la fenêtre. L'état du réseau est caracterisé par les pertes de paquets. Il est important de noter que si ces pertes se produisent trop tôt, cela va causer une détérioration des performances de TCP car cela va empêcher la fenêtre de TCP d'atteindre un seuil élevé afin de permettre une bonne utilisation du lien de transmission.

Le modèle utilisé est décrit par la figure 2. Nous considérons une source TCP qui émet des paquets dans un buffer de taille $B$. Les différents paramètres de ce modèle sont:
$\lambda$ : taux de trafic exogène.
$W_{th}$ : valeur du seuil de slow start (ssthresh).
$\mu$ : taux de service de la file d'attente.
$\tau$: délai aller/retour du paquet.
$T$ : temps de séjour d'un paquet dans un système vide. $T = \tau + \mu^{-1}$
$W_{loss}$ : la taille de la fenêtre à la fin de la phase de congestion avoidance (juste après une perte).
$W_{max}$ : la taille maximum de la fenêtre à la fin de la phase de congestion avoidance.
$W_b$ : la taille de fenêtre qui cause une perte de paquet.

Les algorithmes de contrôle de congestion déterminent comment TCP doit éviter et réagir à la congestion. Les deux variables principales sont la fenêtre de congestion "cwnd", et le seuil de slow start : slow start threshold "ssthresh". A l'établissement de la connexion cwnd est initialisée à un paquet de taille MSS[1] et ssthresh à 64KB. On a étudié l'évolution de la taille de la fenêtre en fonction du temps et on a remarqué que les variations de la taille de la fenêtre sont

1. Maximum Segment Size : taille de segment maximum

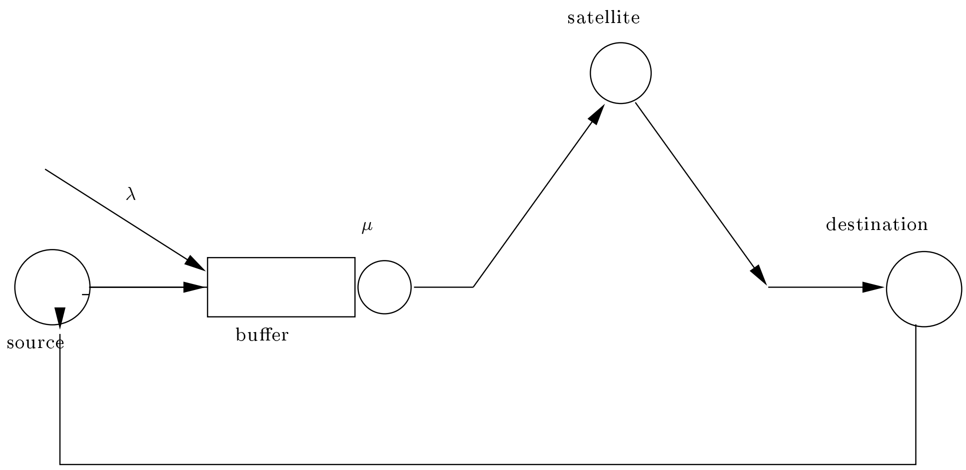


FIG. 2.1: *Modèle TCP*

cycliques. Il n'y a que deux évolutions possibles ou il n 'y a qu'une seule phase slow start par cycle ou deux phases slow start par cycle. Il peut arriver d'avoir trois phases de slow start par cycle mais cela reste rare.

## 2.1 Une seule phase slow start par cycle

Chaque cycle est composé de deux phases : une première exponentielle où à chaque acquittement reçu deux paquets sont envoyés, la deuxième phase est linéaire : la taille de la fenêtre est incrémentée d'un paquet par RTT.

L'évolution de la taille de la fenêtre est la suivante : quand une perte de paquet est détectée, la fenêtre est réduite à un paquet. Dans la phase slow start qui suit, la fenêtre va augmenter rapidement jusqu'à ce qu'elle atteigne la moitié de la valeur à laquelle la perte a été détectée. On passe alors à la phase de congestion avoidance. Durant la phase exponentielle la taille de la fenêtre augmente rapidement jusqu'à ce qu'elle atteigne le seuil "ssthresh", à partir duquel la fenêtre va augmenter plus lentement jusqu'à la détection d'une perte de paquet qui marque la fin d'un cycle.

## 2.2 Deux slow start par cycle

Une première perte de paquet a lieu durant la phase de slow start, la fenêtre de congestion est mise à un et le seuil de slow start "ssthresh" est réduit de moitié. Une seconde phase de slow start commence jusqu'au nouveau seuil "ssthresh" où on passe à la phase de congestion avoidance, à la fin de la phase de congestion avoidance il se produira une perte de paquet, qui marquera la fin de ce cycle.

## 2.3 Quand perd-on un paquet dans la phase slow start?

On voudrait déterminer sous quelle condition une perte se produit durant la phase slow start. Une étude a été réalisée sans trafic exogène, avec et sans perte aléatoire, elle est décrite dans [Lak97]. Une deuxième étude qui a considéré le trafic exogène est décrite dans [EA95].

On définit la taille du buffer normalisée comme étant:

$$\beta = \frac{B}{\tau(\mu - \lambda) + 1} = \frac{B}{\mu T} \tag{2.1}$$

On s'intéresse aux réseaux à produit délai par bande passante élevé donc aux réseaux tels que $\beta \leq 1$ i.e. la taille du buffer est petite comparée au produit.

La taille maximale de fenêtre qui peut être atteinte est

$$W_{pipe} = \mu T + B \tag{2.2}$$

le buffer est alors occupé, et il y a $\mu T$ paquets en circulation.

Si la taille relative du buffer n'est pas assez grande, on a des pertes durant slow start.

On voudrait connaître la taille de la fenêtre quand se produit cette perte de paquet.

Sans trafic exogène: durant la phase slow start et tant que la file d'attente n'est pas pleine : la réception d'un acquittement implique la transmission de deux paquets, on peut en déduire que le taux de transmission des paquets est de $2\mu$.

La file d'attente se construit à un taux $\mu = 2 \cdot \mu - \mu$ .

Avec trafic exogène de taux $\lambda < \mu$: le serveur est partagé entre la source TCP et le trafic exogène; les acquittements arriveront à un taux inférieur à $\mu$ donc les bursts de paquets auront une vitesse inférieure à $2 \cdot \mu$.

Avant de noter la perte du premier paquet il va s'écouler un temps de $\frac{B}{\mu}$, en supposant que la taille du burst est suffisamment grande. Au temps $\frac{B}{\mu}$ le nombre de paquets provenant de la source TCP durant le burst est: $W_b = (2\mu - \lambda)\frac{B}{\mu}$. $(2\mu - \lambda)$ étant le taux de trafic d'entrée dans la file d'attente.

Donc une perte se produit durant slow start si la valeur de la fenêtre est inférieure au seuil "ssthresh" ou $W_{th}$, i.e.

$$W_b \leq W_{th} \tag{2.3}$$

Les paramètres suivants ont été calculés dans [EA95]:

$$W_{max} = W_{loss} + 2 \approx W_{loss}. \tag{2.4}$$

$$W_{max} = (\frac{B}{\mu} + T)(\mu - \lambda) + \frac{\lambda}{\mu}. \tag{2.5}$$

$$W_{max} = B(1 - \frac{\lambda}{\mu}) + T(\mu - \lambda) + \frac{\lambda}{\mu}. \tag{2.6}$$

$$W_b = (2\mu - \lambda)\frac{B}{\mu}. \tag{2.7}$$

$$W_b = (2 - \frac{\lambda}{\mu})B. \tag{2.8}$$

## 2.4 Analyse des améliorations proposées

### 2.4.1 Première proposition

Dans TCP Tahoe la valeur du ssthresh est définie par:

$$W_{th} = \frac{W_{max}}{2}$$

notre proposition est de prendre comme valeur de ssthresh:

$$W_{th} = \alpha W_{max}.$$

La valeur de $\alpha$ serait comprise entre 0 et 1.

On reprend la condition pour avoir une perte durant slow start :

$$W_b \leq W_{th} \tag{2.9}$$

on pose

$$W_{th} = \alpha W_{max} \text{ avec } 0 \leq \alpha \leq 1 \tag{2.10}$$

alors

$$\alpha^{-1} W_b \leq W_{max} \tag{2.11}$$

$$\alpha^{-1}(2 - \frac{\lambda}{\mu})B \leq B(1 - \frac{\lambda}{\mu}) + T(\mu - \lambda) + \frac{\lambda}{\mu} \tag{2.12}$$

$$\alpha^{-1}(2 - \frac{\lambda}{\mu})B - B(1 - \frac{\lambda}{\mu}) \leq T(\mu - \lambda) + \frac{\lambda}{\mu} \tag{2.13}$$

$$B(\frac{2}{\alpha} - \frac{\lambda}{\alpha\mu} - 1 + \frac{\lambda}{\mu}) \leq (\tau + \mu^{-1})(\mu - \lambda) + \frac{\lambda}{\mu} \tag{2.14}$$

$$B(\frac{2\mu - \lambda - \alpha\mu + \alpha\lambda}{\alpha\mu}) \leq (\tau + \mu^{-1})(\mu - \lambda) + \frac{\lambda}{\mu} \tag{2.15}$$

$$\frac{B(\mu(2 - \alpha) + \lambda(\alpha - 1))}{\alpha\mu} \leq \tau(\mu - \lambda) + \mu^{-1}\mu \tag{2.16}$$

$$\frac{B}{\tau(\mu-\lambda)+1} \leq \frac{\alpha\mu}{\mu(2-\alpha)+\lambda(\alpha-1)} \tag{2.17}$$

$$\frac{B}{\tau(\mu-\lambda)+1} \leq \frac{\alpha}{(2-\alpha)+\frac{\lambda}{\mu}(\alpha-1)} \tag{2.18}$$

par définition

$$\beta = \frac{B}{\tau(\mu-\lambda)+1} \tag{2.19}$$

donc

$$\beta \leq \underbrace{\frac{\alpha}{(2-\alpha)+\frac{\lambda}{\mu}(\alpha-1)}}_{\beta_{max}} \tag{2.20}$$

une perte se produit durant slow start avec un trafic exogène nul i.e: $\lambda = 0$ si

$$\beta = \frac{B}{\tau\mu+1} \leq \frac{\alpha}{2-\alpha} \tag{2.21}$$

$$\tau\mu + 1 \geq B\frac{2-\alpha}{\alpha} \tag{2.22}$$

en prenant

$$\alpha = \frac{1}{2} \tag{2.23}$$

on retouve la condition

$$\tau\mu + 1 \geq 3B \text{ et donc} \tag{2.24}$$

$$\beta \leq \frac{1}{3} \tag{2.25}$$

et avec un trafic exogène non nul de taux $\lambda$ :

$$\beta \leq \underbrace{\frac{\alpha}{(2-\alpha)+\frac{\lambda}{\mu}(\alpha-1)}}_{\beta_{max}} \tag{2.26}$$

On veut borner la valeur de $\beta_{max}$. Il ne faut pas que sa valeur soit trop grande car à ce moment là, la probabilité d'avoir une perte durant slow start va augmenter, ni trop petite car cela signifierait que la valeur de $\alpha$ est très petite et cela induira une valeur de fenêtre de congestion au début de la phase de congestion avoidance très faible et l'on n'atteindra pas la capacité du lien. La valeur d'alpha choisie ne doit pas induire une taille de fenêtre de congestion plus petite que celle qu'on aurait obtenue avec deux phases de slow start et donc la valeur d'$\alpha$ ne peut pas être plus petite que $\frac{1}{4}$.

### 2.4.2 Deuxième proposition

Si l'on observe une perte de paquets durant la phase slow start, c'est à cause de l'émetteur qui envoie plus de paquets qu'il n'en faudrait. A un moment donné, l'émetteur dépasse la vitesse limite. Il a été proposé plusieurs méthodes pour réduire le débit de l'émetteur mais qui serait effectué du côté du récepteur (ou des routeurs intermédiaires). Nous proposons de modifier l'algorithme de l'émetteur pour qu'il n'envoie pas de bursts trop violents. Il faut pour cela définir une vitesse limite, durant la phase slow start l'émetteur ne double pas forcément sa vitesse à la réception d'un acquittement. S'il doit envoyer deux paquets, il observera un délai entre l'envoi de ces deux paquets afin de ne pas déborder la file d'attente avec un burst durant toute la phase de slow start.

## 2.5 Etude de la fonction $\beta_{max}$ en fonction de $\rho$

On définit $\rho$ comme étant $\rho = \frac{\lambda}{\mu}$.
On veut étudier le comportement de $\beta_{max}$ pour des valeurs de $\rho$ variant entre 0 et 1.
Definissons $\hat{\beta}_{max}$:

$$\hat{\beta}_{max} = (1-\rho)\beta_{max} \tag{2.27}$$

$$\hat{\beta}_{max} = \frac{B}{\tau(\mu-\lambda)} = \frac{B}{\tau\mu(1-\rho)} \tag{2.28}$$

et dans le cas particulier où $\lambda = 0$ :

$$\hat{\beta}_{max} = \frac{B}{\tau\mu} \geq \beta_{max} \tag{2.29}$$

on a:

$$\frac{\tau\mu}{B} \leq \frac{1}{(1-\rho)\beta}_{max} \tag{2.30}$$

revenons au cas général:

$$\frac{1}{(1-\rho)\beta}_{max} = \frac{1}{1-\rho} \cdot \frac{(2-\alpha)+\rho(\alpha-1)}{\alpha} \tag{2.31}$$

donc

$$\frac{\tau(\mu-\lambda)}{B} \leq \frac{1}{1-\rho} \cdot \frac{(2-\alpha)+\rho(\alpha-1)}{\alpha} \tag{2.32}$$

En consultant la figure 2.2, qui représente $\frac{1}{(1-\rho)\beta_{max}}$ pour différentes valeurs d'$\alpha$[2].

On voudrait récupérer de la bande passante entre les deux courbes (pour $\alpha = \frac{1}{2}$ et $\alpha = \frac{1}{3}$). Les simulations effectuées valident ce résultat.

2. Nous avons tracé les graphes pour $\alpha$ variant d'$\frac{1}{2}$ à $\frac{1}{8}$. La première courbe en partant du bas étant celle pour $\alpha = \frac{1}{2}$.

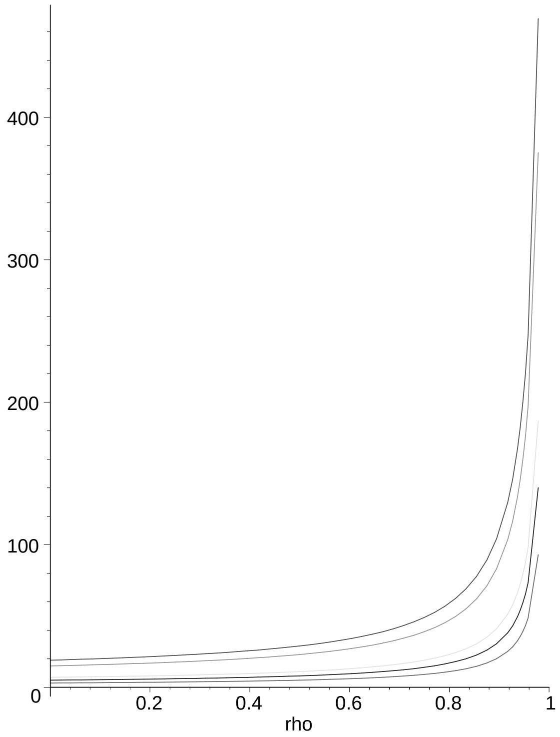


FIG. 2.2: $\frac{1}{(1-\rho)\cdot\beta_{max}}$

## 2.6 Simulations

### 2.6.1 Description des simulations réalisées

Le réseau utilisé pour les simulations est décrit par la figure 2.1. Le rectangle étant un buffer de capacité finie, les ronds représentent la source et la destination. Ces simulations utilisent des routeurs de type (drop-tail) avec des petits buffers. Les valeurs choisies pour la taille du buffer n'étant pas très réalistes, le but étant d'illustrer le comportement des algorithmes de congestion de TCP. Le simulateur utilisé est NS [3]. NS [MF95] est basé sur l'ancien simulateur développé à LBLN tcpsim qui à son tour est basé sur le simulateur REAL. Les graphes des simulations ont été générés en traçant les paquets entrant et sortant du routeur. Pour chaque graphe, l'axe des $x$ représente le temps d'arrivée ou de départ en secondes. L'axe des $y$ représente la taille de la fenêtre. Le but des simulations était de bien montrer qu'il y avait des situations où l'on avait intérêt à choisir $\alpha$ égal à $\frac{1}{3}$ plutôt qu' $\alpha$ égal à $\frac{1}{2}$. Divers simulations ont été réalisées.

Le comportement idéal "sans pertes" durant slow start est représenté par la figure 2.3, nous voyons bien le comportement cyclique de la taille de la fenêtre en fonction du temps [4]. Les paquets ont une taille de 576 octets. Une bande passante de 300 paquets par seconde, un buffer de taille 9 (paquets). Le délai étant de 260 ms.

Nous avons analysé le comportement de la fenêtre TCP en fonction du temps en prenant différentes valeurs pour $\alpha$. On remarque que dans les mêmes conditions, en prenant la valeur d'$\alpha$ égale à 1/3, on gagne en débit.

En effet, dans la figure 2.4 où $\alpha = 1/3$, il n'y a qu'une seule phase de slow start par cycle alors que dans la figure 2.5 où $\alpha = 1/2$, il y en a deux. Les simulations ont été réalisées dans les mêmes conditions i.e. un débit de la source

3. La version de Network Simulator utilisé est la 2.lb2.
4. Une phase de slow start suivie d'une phase de congestion avoidance.

vers la file d'attente de 300 paquets/s, un débit de sortie de 170 paquets/s, un délai de $250ms$, et un buffer de 10 paquets. Les paquets ont une taille de 1024 octets. Le débit moyen de la figure 2.4 qui est de 150 paquets/s, est meilleur que celui de la figure 2.5 qui est égal à 120 paquets/s.

Nous avons également effectué des simulations avec des valeurs d'$\alpha$ plus petites, en particulier pour $\alpha = 1/4$ et là même si nous n'avons pas de pertes durant slow start, nous rentrons trop tôt dans la phase de congestion avoidance, et perdons en débit (figure 2.9) car pour atteindre le débit maximum il faut un temps beaucoup plus long, la phase de congestion avoidance étant plus conservatrice. Le débit moyen de la figure 2.9 est seulement de 206 paquets/s, avec un débit d'entrée dans la file d'attente de 3MB/s.

Il faut noter que si la taille du buffer est grande, nous n'avons pas ce phénomène. Comme par exemple dans les figures 2.6 et 2.7 où la taille du buffer est de 100 paquets, les autres paramètres étant les mêmes que ceux des simulations des figures 2.4 et 2.5. Ici nous n'avons pas de pertes de paquets durant slow start [5]. Le débit moyen de la figure 2.6 est de 117 paquets/s. Alors que dans la figure 2.7, il est de 122 paquets/s. Cependant les réseaux que l'on considère ont une taille de buffer petite comparée au produit délai par bande passante. Nous avons noté durant ces simulations, qu'en général au début de la phase slow start du premier cycle, il y a des pertes et cela même si plus tard (dans les autres cycles), il n'y avait pas de pertes durant la phase slow start.

Pour valider la deuxième proposition qui consiste à espacer l'envoi des données. On a diminué le débit de la source vers la file d'attente. On provoque ainsi une augmentation de l'espacement entre les paquets. On observe le résulat dans la figure 2.8 où le débit de la source [6] est de 550 paquets/s, le taux de service [7] est de 300 paquets/s, un délai de 250ms, et un buffer de 10 paquets. Nous avons comparé ces résulats avec ceux obtenus dans les mêmes conditions mais avec un débit d'entrée dans la file d'attente beaucoup plus important, il était de $3MB/s$. Dans la dernière configuration, nous avons des pertes durant slow start, alors que dans la figure 2.8, les pertes (durant slow start) ont disparu. Le débit moyen de la figure 2.8 est de 226 paquets/s.

### 2.6.2 Résultats

En conclusion et étant donné que le phénomène de pertes durant slow start est courant dans les réseaux à produit délai par bande passante élevé, le seuil de ssthresh devrait être $W_{th} = \frac{W_{max}}{3}$. L'utilsateur devrait pouvoir choisir la valeur d'$\alpha$ qui lui convient (selon l'environnement). De plus sachant que les causes des pertes durant slow start sont dues à des bursts trop violents, les envois de paquets devraient être espacer.

5. Ce qui confirme le résultat obtenu car ici $\beta > \frac{1}{3}$ pour $\alpha = \frac{1}{2}$ et $\beta > \frac{1}{5}$ pour $\alpha = \frac{1}{3}$.
6. Le débit d'entrée dans la file d'attente.
7. Le débit de sortie de la file d'attente.

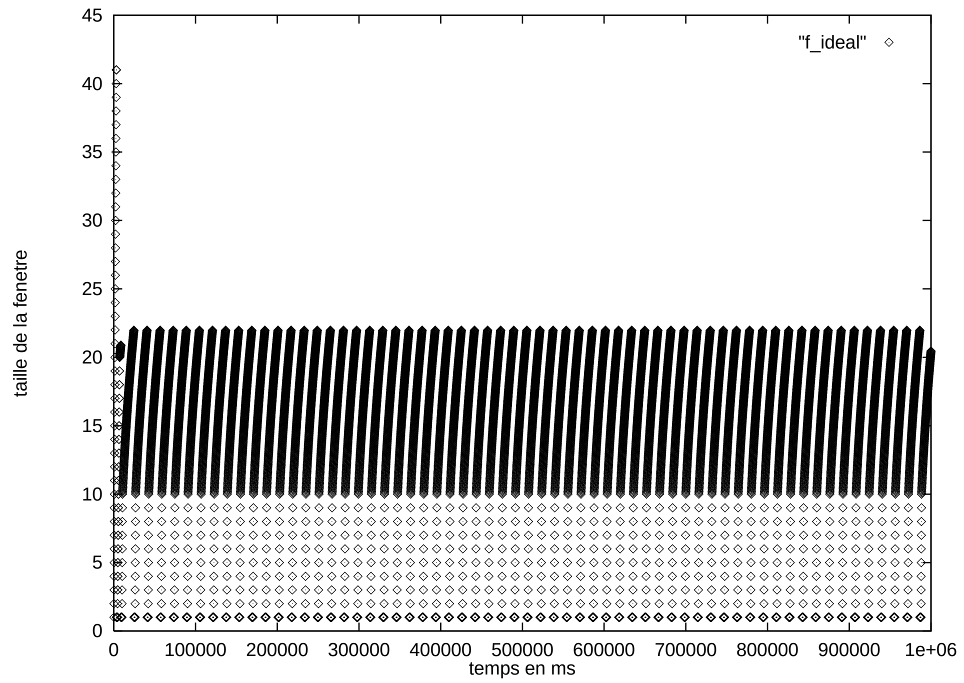


FIG. 2.3: *Comportement idéal de la fenêtre avec* $\alpha = \frac{1}{2}$

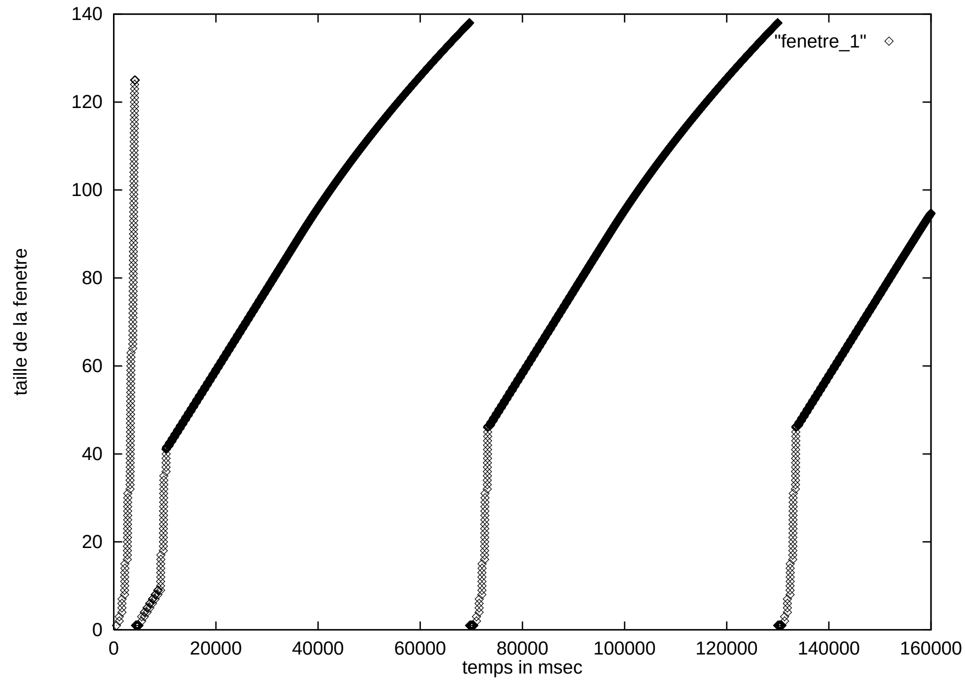


FIG. 2.4: *Une phase de slow start par cycle avec* $\alpha = \frac{1}{3}$

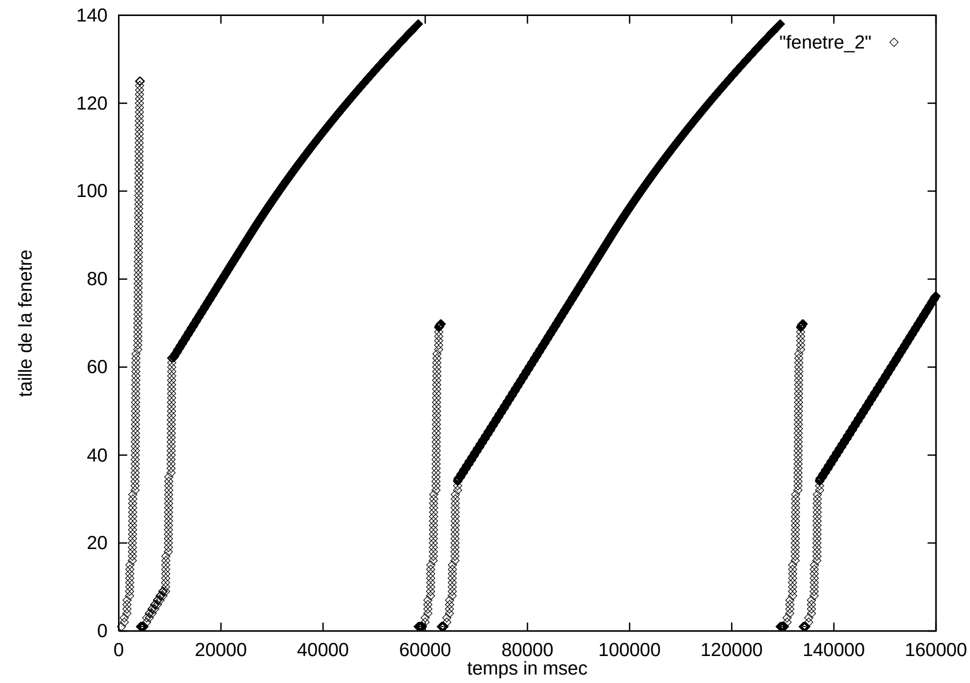


FIG. 2.5: *Deux phases de slow start par cycle avec* $\alpha = \frac{1}{2}$

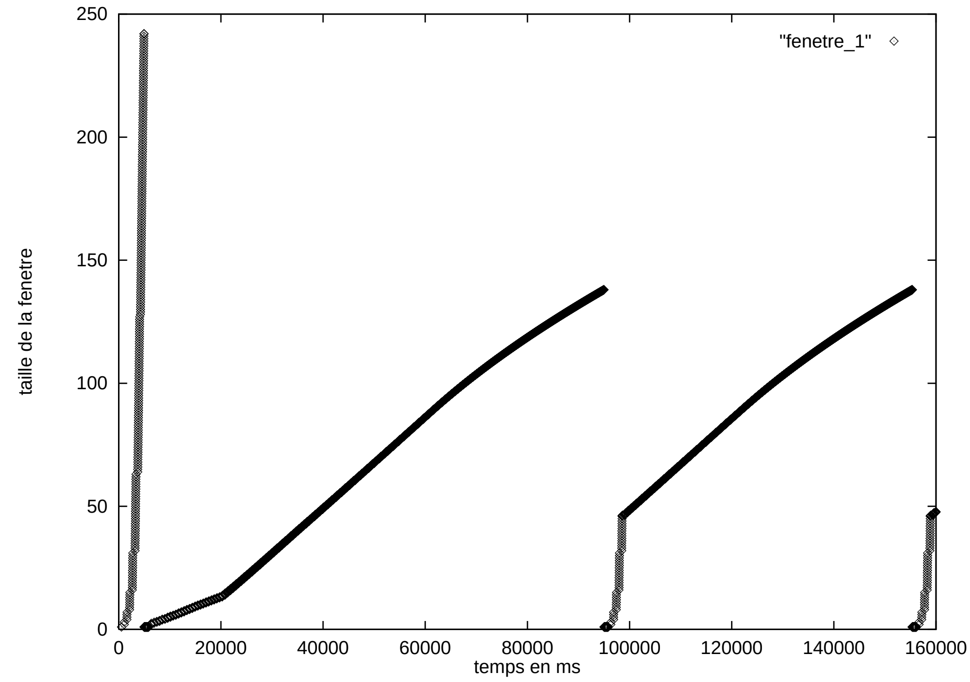


FIG. 2.6: *Comportement de la fenêtre avec* $\alpha = \frac{1}{3}$

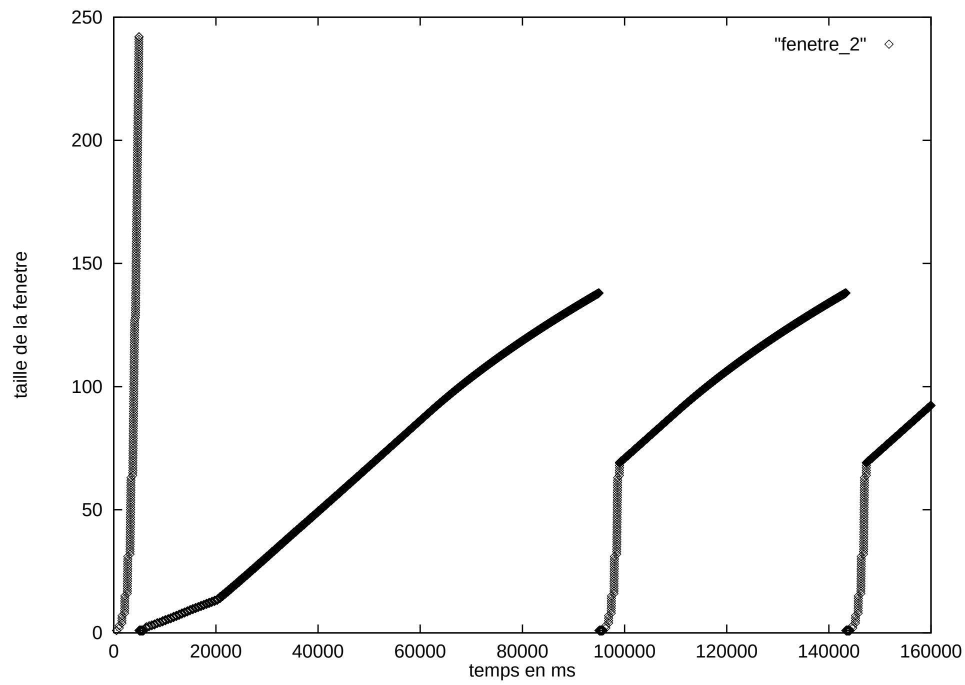


FIG. 2.7: *Comportement de la fenêtre avec* $\alpha = \frac{1}{2}$

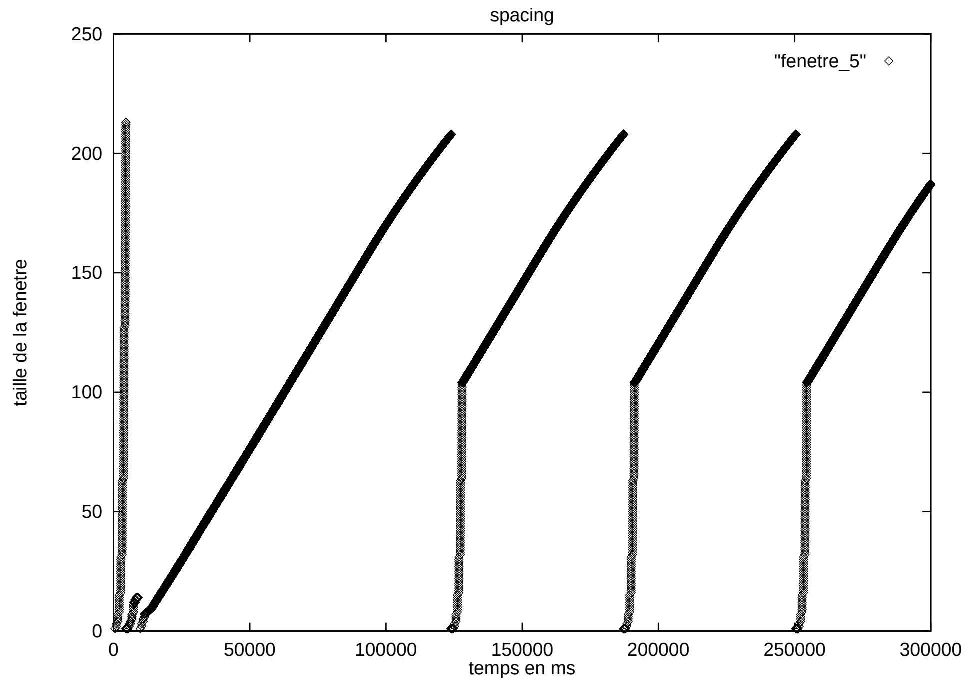


FIG. 2.8: *Comportement de la fenêtre avec une bande passante plus faible.*

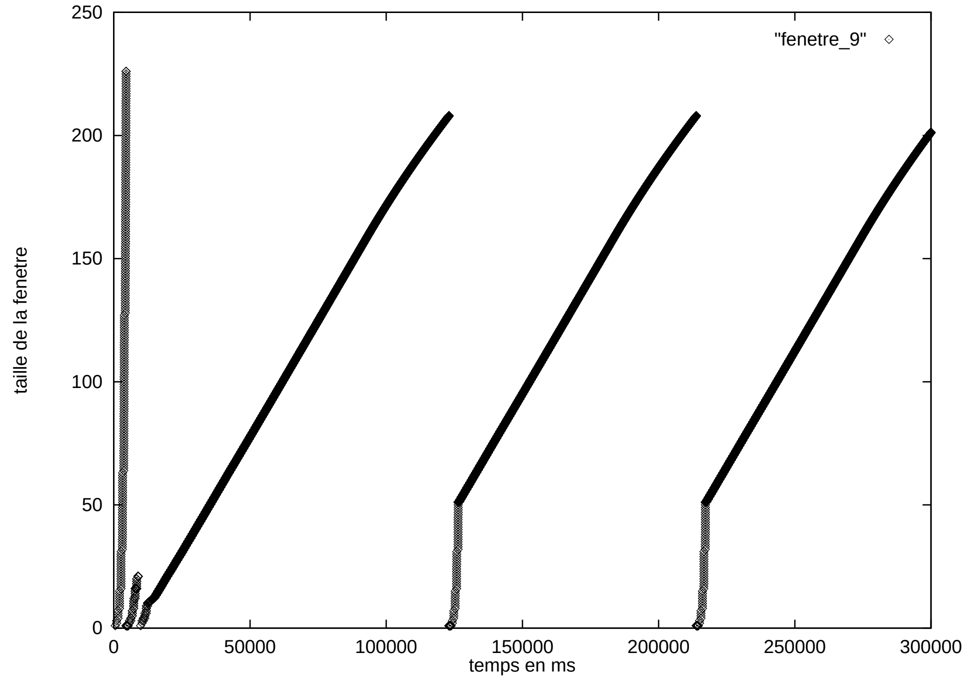


FIG. 2.9: *Comportement de la fenêtre avec* $\alpha = \frac{1}{4}$.

# Troisième partie

# Contrôle d'erreur

# Chapitre 1

# Codes correcteurs d'erreurs

Un système de communication est representé par la figure 1.

L'encodeur transforme les données en entrée sous forme d'alphabet binaire en signaux acceptables par le canal de transmission (modulation). Les canaux de transmission ont une capacité limitée et si le débit de la source est inférieur à cette capacité il est possible de choisir un ensemble de signaux de sortie de telle sorte que la probabilité de décodage erroné soit arbitrairement faible (Théorème de Shanon).

Cette théorie ne dit pas comment construire cet ensemble de signaux et ne garantit pas qu'un tel système puisse être conçu.

L'encodage pour les codes correcteurs n'est pas plus compliqué que la détection d'erreur, c'est le décodage qui requiert un équipement plus important chez le récepteur. Pour une transmission utilisant un lien satellite où l'équipement de l'émetteur est plus coûteux que celui du récépteur, on utilise des procédures de correction d'erreurs. Dans d'autres situations, il peut être préférable d'utiliser la détection d'erreur puis la retransmission, la détection étant plus simple par nature que la correction et nécéssite un équipement de décodage plus simple. On distingue deux types de corruption de données: une erreur qui est un bit qui a une valeur inconnue à une location inconnue. un effacement (erasure) est un bit qui a une valeur inconnue à une location connue. Il y a deux types de codes fondamentaux :

## 1.1 Code en bloc

L'encodeur d'un code en bloc doit partitionner la suite d'informations en entrée en blocs ou sections de k symboles, il doit ensuite opérer sur ces blocs

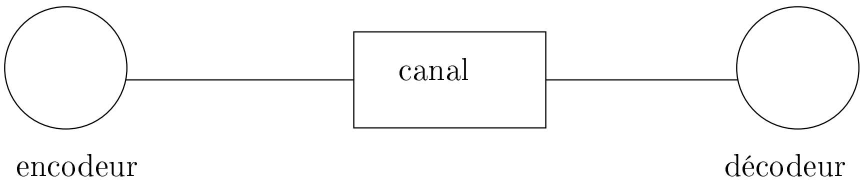


Fig. 1.1: *Schéma de codage*

indépendamment suivant le code particulier qu'il emploie. A chaque bloc d'informations est associé un $n$-uple de symboles ($n > k$) qui forme un mot de code ($n$ est la longueur du code ou du bloc). On adjoint aux $k$ symboles d'informations un certain nombre $(n-k)$ de symboles calculés en fonction des premiers par l'intermédiaire d'une fonction $\Psi$ fixée à l'avance. Ces $(n-k)$ symboles forment un bloc que l'on appelle redondance ou symboles de contrôle. Le mot sera transmis, corrompu par le bruit et décodé à la réception indépendamment de tous les autres mots.
Une définition plus mathématique: On appelle code en bloc de longueur $n$ sur $F$ [GC92] un sous-ensemble C de $F^n$. $F$ étant un corps fini. La cardinalité $q = |F|$ est en général une puissance d'un nombre premier $p$. Les paramètres d'un code en bloc sont :

- La longueur des mots du code $n$.
- Cardinal ou dimension du code $k$ qui est par définition

$$k = log_q(C). \tag{1.1}$$

  $k$ représente le nombre de symboles d'informations par mot de code.
- Taux de transmission ou de rendement du code qui est défini par $\frac{k}{n}$.
- Distance minimale.

Comme exemple il y a les codes de parité, codes à répétition et codes de Hamming. La distance minimale (de Hamming) indique la capacité de détection et de correction du code.

## 1.2 Code en arbre

Le codage en arbre opère sur la suite d'informations sans la découper en blocs indépendants.

Un code en arbre associe une séquence de code (qui peut être infinie) à une séquence d'informations. Les lois de cette association sont décrites à l'aide d'un graphe (arbre). On utilise la même terminologie que pour les codes en blocs.

Etant donnée une séquence en entrée l'arbre est utilisé de la manière suivante : tout d'abord l'information $q$-aire (en général $q = 2$) est découpée en blocs de $k0$ symboles: i.e. si l'on code la séquence symbole par symbole, on les prendra un élément à la fois ($k0 = 1$), une convention est utilisée pour savoir quelle branche de l'arbre sera prise, un chemin est tracé dans cet arbre. L'ensemble des ($n0$-uples) associés aux chemins forment le code spécifié par la séquence d'informations. Le taux du code est donné par $\frac{k}{n0}$. A la réception, l'opération inverse est effectuée; le décodage consiste à bien deviner la séquence émise, corrompue par le canal de transmission. Le décodeur en bloc ne doit examiner que $n$ symboles reçus pour savoir quelle était la séquence émise, alors que le décodeur en arbre peut avoir à traiter une très longue séquence, certaines restrictions sont utiles. Il est supposé que le récepteur ne peut traiter qu'un nombre fixe $m$ de symboles reçus (composé de sections de $n0$ symboles), la quantité $m \cdot n0 = n$ est appelée la contrainte de longueur du code. De plus le décodeur ne peut pas revenir en arrière : quand un symbole quitte le décodeur il est perdu, le décodage s'effectue une branche à la fois ($n0$-uples à la fois).

### 1.2.1 Code convolutionnel

Les codes convolutionnels[1] forment une classe importante de codes utilisés par les ingénieurs, c'est un sous ensemble de la classe des codes en arbre et sont plus simples à implémenter que tous les autres codes en arbre. Le processus de codage se déroule d'une façon continue : on considère des séquences de symboles d'information et des séquences codées de longueur indéfinie. Un code convolutionnel $[n, k]$ est généralement défini à partir d'un codeur, c'est à dire d'un circuit séquentiel linéaire à $k$ entrées et $n$ sorties. Le but étant de trouver, pour un code donné, un codeur avec un nombre minimal de mémoires.

Le théorème de Shannon s'applique sur les deux types de codes: ils ont la même limitation et les mêmes possibilités.

Les codes utilisés avec les ARQ[2] [SL84] sont les CRC[3] qui assurent une transmission fiable mais il faut noter que la fiabilité n'est pas le seul critère à considérer. En effet, le rendement chute à moins de 50% dès que le taux d'erreur est supérieur à $10^{-4}$. Pour y remédier, les systèmes hybrides ont été conçus (ils utilisent à la fois l'ARQ et une correction directe d'erreurs) et ont un meilleur rendement que l'ARQ seul.

Lorsqu'on veut un système de codage puissant on peut envisager deux ou plusieurs niveaux de codage. On code d'abord l'information avec un code $C1$ (code externe). On passe d'une suite de longueur $k$ à une suite codée de longueur $n$. Cette nouvelle séquence (de longueur $n$) est alors considérée comme de l'information pour un second code $C2$ (code interne). Généralement le code externe est plus spécialement construit pour lutter contre les erreurs engendrées par le code interne lorsqu'il se trompe. On obtient ainsi des systèmes de codage très puissants au prix d'une électronique plus lourde que pour les systèmes élémentaires. Exemple : En utilisant un système concaténé composé d'un code RS[4] (externe) et d'un code de Hamming (interne). Dans ce cas on obtient un taux résiduel de $10^{-9}$ à partir d'un taux d'entrée de $2 \cdot 10^{-2}$. Les codes de Reed Solomon sont adaptés aux coupures alors que les codes de Hamming luttent contre les erreus isolées. Un gain de codage de $5dB$ est possible à $10^{-5}$ avec un code interne de taux $\frac{1}{2}$ pour une complexité bien moindre qu'en utilisant un code convolutionnel comme code interne. L'utilisation d'un code de Hamming comme code interne, conduit à une complexité électronique faible.

Les systèmes concaténés RS plus un code en bloc :

- RS + code orthogonal. Une caractéristique de cette classe de codes concaténés (code RS comme code externe) est qu'ils ont un faible taux de transmission. Ils sont utilisables uniquement quand on dispose d'une largeur de bande suffisante. Mais par contre ils ne nécessitent qu'une faible puissance.

- RS + code court. On obtient un taux modéré $1/4 \leq R \leq 3/4$ qui est un avantage sur le système précédent. L'autre avantage est la possibilité de hauts débits pouvant atteindre 40 Mbits/s.

1. La terminologie code convolutionnel est une traduction de l'anglais "convolutionnal code". On parle aussi de codes récurrents on convolutifs.
2. Automatic Repeat reQuest.
3. Cyclic Redundancy Code.
4. Reed Solomon.

# Chapitre 2

# Mécanisme de la FEC

Le mécanisme de correction d'erreurs utilisé pour assurer une transmission fiable sur les liens satellites est le Forward Error Correction. Cette technique consiste à transmettre des informations redondantes, si une partie des données est perdue on peut la reconstruire grâce à ces informations redondantes, cela fournit une fiabilité sans augmenter le temps de latence. Compte tenu du délai imposé qui est au minimum de l'ordre de 500ms, cela nous évite d'avoir à retransmettre les paquets erronés. La quantité d'informations redondantes doit rester faible afin que la FEC reste efficace. Quand la FEC est utilisée, deux effets antagonistes apparaissent:

- la quantité d'informations redondantes dues à la FEC corrige une partie des pertes (erreurs).
- Les données ajoutées dues à la FEC augmentent le taux de pertes.

La FEC reste efficace si le premier point l'emporte sur le second. Le potentiel de la FEC à corriger les erreurs (pertes) dépend du comportement des pertes dans le réseau. Donc pour pouvoir proposer des mécanismes efficaces de redondance il faut d'abord effectuer des mesures pour caractériser les pertes sur le lien satellite.

## 2.1 Les caractéristiques des erreurs sur le lien satellite

Si bien configurés, les liens satellites exhibent une caractéristique On/Off similaire à la fibre optique. Le standard DVB[1] utilise la FEC concaténée pour assurer une transmission fiable; sans erreur sur le lien satellite. La FEC est composée de deux codes:

- Le code externe qui est un code en bloc de type Reed Solomon de paramètres (204,188,8) et qui peut corriger 8 octets erronnés sur un bloc de 204 octets: 188 utiles + 16 octets de redondance.
- Le code interne qui est un code convolutif de contrainte de longueur 7.

1. Direct Video Broadcasting.

Le taux de FEC le plus utilisé est de 3/4 : un octet redondant tous les trois octets utiles.
Le schéma simplifié :
octets utilisateurs $\Rightarrow$ RS (code externe) $\Rightarrow$ Code Convolutif (code interne) $\Rightarrow$ lien satellite (bruit ajouté) $\Longrightarrow$ Code Convolutif (décodage interne) $\Rightarrow$ RS (décodage externe) $\Rightarrow$ octets utilisateurs reconstruits.

Grâce à cette FEC, on obtient des taux d'erreurs très faibles.
Le "BER raw" est défini comme étant le BER avant le décodage par la FEC interne (après la démodulation).
Avec un Raw BER de $2 \cdot 10^{-2}$, on obtient un BER de $2 \cdot 10^{-4}$ après le décodage par la FEC interne et un BER $< 1 \cdot 10^{-11}$ après le décodage de la FEC externe.

Avec un Raw BER de $1.6 \cdot 10^{-2}$ on obtient un BER de $7 \cdot 10^{-5}$ après le décodage de la FEC interne et un BER $< 1 \cdot 10^{-15}$ après le décodage de la FEC externe.

Quand le Reed Solomon ne peut pas décoder, il va produire de longues chaines d'erreurs, de plus on perd la synchronisation bloc, cela se produit quand le taux d'erreur BER avoisine les $5 \cdot 10^{-2}$. On parle dans ce cas d'évènements d'erreurs plutôt que de taux d'erreurs (BER) au niveau utilisateur.

A EUTELSAT: le lien satellite est dimensionné de façon à ce que le BER ne dépasse pas $5 \cdot 10^{-2}$.

Quand le signal s'affaîblit (à cause des conditions météo : orages, pluie ... ), et qu'il passe en dessous d'un certain seuil, même d'une valeur d'$1dB$, le lien est complètement corrompu et quand le signal passe d'une valeur de 0.5 au dessus du seuil, il redevient totalement fiable.

Les puces qui implémentent le décodage FEC incluent aussi un démodulateur QPSK [2], qui peut traiter un débit allant jusqu'à 60 Mbits/s. Elles offrent plusieurs taux de FEC interne automatiquement détectées.

## 2.2 Etude des pertes sur le lien satellite

On voudrait avoir une modélisation du processus de pertes sur le lien satellite en paramétrant le taux de FEC. L'étude du processus de pertes s'effectue à deux niveaux : niveau applicatif ou niveau paquet et niveau physique car on s'intéresse aux erreurs bits qui reflètent l'état de la liaison.

## 2.3 Importance de cette modélisation

Les pertes aléatoires sont la cause d'une détérioration des performances de TCP. Si on ne veut pas sous utiliser le lien il doit avoir un taux d'erreur suffisamment faible afin de ne pas affecter le comportement de TCP qui interprète toute erreur de transmission comme un signal de congestion. Ce taux doit être de l'ordre de $10^{-12}$ (ce qui est proposé dans les bandes Ka [3]) ou mieux. Pour atteindre ce taux on doit assurer la correction d'erreurs par des moyens de FEC. En connaissant le comportement des erreurs sur le lien satellite on pourra adapter la quantité de FEC pour la rendre optimale. Actuellement le taux utilisé

2. Quadrary Phase Shift keying.
3. bande Ka : fréquence des porteuses du lien montant est 30 GHz et du lien descendant 20 GHz.

est élevé ce qui assure une fiabilité proche de celle de la fibre optique mais en contrepartie on perd de la bande passante.

## 2.4 Le bruit et les erreurs

Lors de la transmission il se produit des erreurs dues à des phénomènes perturbateurs sur le canal: le bruit. Il y a deux types de bruit: le premier qui perturbe brièvement la transmission et qui donne des erreurs isolées, et le second qui perturbe plus longtemps la transmission et qui engendre des coupures (bursts).

Principales origines du bruit:

- Interférence entre symboles : le couple modulateur/démodulateur est la source de ce bruit.
- Diaphonie : phénomène constaté sur des transmissions par câbles. Plusieurs liaisons sont acheminées simultanément dans un même câble, et cela induit l'interférence d'une transmission sur une autre.
- Saut et scintillation de phase : car lors de la transmission d'un signal il peut être soumis à des variations de phases de deux types : brusques (sauts de phases) ou lentes causées par l'alimentation éléctrique d'élements constituant le système de transmission.
- Micro-coupure : ce sont de courtes interruptions du signal, d'une durée de quelques millisecondes. Il faut noter qu'une microcoupure d'une durée de $1ms$ pour une transmission à $10Mbits/s$ perturbe 10000 symboles binaires, ce qui est considérable.
- Bruit galactique : cela est dû au rayonnement des diverses sources d'énergie de l'espace, il engendre le bruit blanc.

La qualité du lien satellite peut aussi être dégradée à cause des effets de l'atténuation et de distortion. L'atténuation est la perte d'énergie durant la propagation et dépend de la fréquence, distance et conditions météo. On décrit la qualité du lien satellite par le rapport signal/bruit ou $S/N$.
Le résultat de la transmission d'un code (0 ou 1) peut être modélisé par une variable aléatoire gaussienne de moyenne $\mu$ et d'écart type $\sigma$. Si $E_s$ est l'énergie par symbole binaire transmis, et $N_0$ la densité spectrale du bruit. On a $\mu = \sqrt{N_0}/2$. Plus $E_s$ est grande (plus on apporte de l'énergie par symbole) plus le signal est à l'abri du bruit. Plus $N_0$ est grand (i.e. plus le niveau du bruit est élevé) plus la liaison est mauvaise.

## 2.5 Influence de la pluie sur l'état de la liaison satellite

D'après des mesures correspondant au climat européen, la présence de la pluie cause l'atténuation du signal émis d'une valeur approximative de $10dB$ sur le lien montant, et de $8dB$ sur le lien descendant [Fri95] et [Jun97], avec des fréquences de $14GHz$ et $12GHz$ respectivement. Cette atténuation diminue la valeur de $\frac{C}{N_0}$. $\frac{C}{N_0}$ étant la densité spectrale du signal.

## 2.6 Modélisation du canal

On cherche à modéliser les erreurs induites par le bruit. Cela peut servir à valider un système de codage. Il y a plusieurs modèles théoriques qui représentent la distribution des erreurs.

- modèle de Poisson: ici on suppose que les intervalles entre deux erreurs consécutives sont indépendants, et suivent une loi exponentielle de paramètre $\lambda$. La variable aléatoire $X$ associée au nombre d'erreurs par unité de temps suit une loi de Poisson de paramètre $k$, donnée par :

$$P(X = k) = e^{-\lambda \frac{\lambda^k}{k!}}, E(X) = \lambda, \sigma = \sqrt{k}.$$

- Modèle de Neymann-Pearson: ce modèle a été utilisé en 1981 pour modéliser les canaux à bursts. L'arrivée des bursts suit une loi de POISSON de paramètre $m1$, et le nombre d'erreurs dans les bursts suit une loi de POISSON de paramètre $m2$.

Pour modéliser les erreurs sur le lien satellite, nous avons pensé au modèle MMPP, qui est utilisé pour modéliser les corrélations, car d'après les tests effectués, les erreurs se produisent en burst.

## 2.7 Description des tests réalisés

On s'est intéressé aux délais inter-pertes. Nous avons effectué les premiers tests avec le taux de FEC fixé à $\frac{3}{4}$ qui est le taux le plus utilisé[4]. Les résultats n'ont montré aucune perte. Le taux de FEC a été réduit à $\frac{7}{8}$ puis à $\frac{8}{9}$ dans le but d'observer des pertes de paquets. Les résultats n'ont toujours pas montré de pertes. Finalement des pertes de paquets ont été obtenues en baissant la puissance d'émission de l'antenne, et lorsque les conditions atmosphériques étaient défavorables.

## 2.8 Définition du processus MMPP

Un processus Markov Modulated Poisson Process [WF93] est un processus stochastique avec un taux d'arrivée défini par $\lambda^*[J(t)]$. $J(t)t \geq 0$ étant un processus de Markov à $m$ états irréductibles. La particularité de ce processus est que quand il se trouve dans l'état $i$, les arrivées suivent un taux $\lambda_i$. Les taux d'arrivées du processus varient aléatoirement avec le temps. Ce modèle capture les corrélations entre les temps d'inter-arrivées. Pour définir la matrice de transition du processus de Markov, on pose $J_0$ l'état de $J(t)$ à l'instant $t = 0$ et $X_0 = 0$. A la $k^{ieme}$ arrivée du MMPP, on associe l'état correspondant $J_k$ du processus de Markov ainsi que le temps $X_k$ entre la $k^{ieme}$ arrivée et la $(k-1)^{ieme}$ arrivée. La matrice de transition de la séquence $[(J_n, X_n), n \geq 0]$ est définie par :

$$F(x) = \int_0^x e^{(Q-\Lambda)u} \, du \, \Lambda = \left[I - e^{(Q-\Lambda)x}\right] (\Lambda - Q)^{-1} \Lambda.$$

4. 1 octet redondant tous les trois utiles.

Q étant le générateur infinitésimal de la chaine de markov à m états continue. $\Lambda$ est la matrice des taux d'arrivées. Les moments entre les temps d'arrivées peuvent être obtenus en différentiant les transformées de Laplace. La matrice de la transformée de Laplace est donnée par:

$$f^*(s) = \int_0^\infty e^{-sIt} F(\mathrm{d}t) = \int_0^\infty e^{-sIt} e^{(Q-\Lambda)t} (\Lambda - Q)(\Lambda - Q)^{-1}\Lambda \, \mathrm{d}t$$
$$= \int_0^\infty e^{-(sI-Q+\Lambda)t} \, \mathrm{d}t \, \Lambda = (sI - Q + \Lambda)^{-1}\Lambda.$$

Ces moments peuvent être obtenus en choisissant les conditions initiales adéquates. En posant $T_{A,i}$ comme étant le temps entre la $i^{ieme}$ et la $(i+1)^{ieme}$ arrivée de la version des temps d'arrivées stationnaires du processus MMPP. On obtient:

$$\begin{aligned} E[T_{A,i}^k] &= k!p[(\Lambda - Q)^{-1}\Lambda]^{i-1}(\Lambda - Q)^{-(k+1)}\Lambda e \\ &= k!p(\Lambda - Q)^{-(k+1)}\Lambda e. \end{aligned} \tag{2.1}$$

et

$$\begin{aligned} E[(T_{A,1} - &E[T_{A,1}])(T_{A,(k+1)} - E[T_{A,(k+1)}])] = \\ &p(\Lambda - Q)^{-2}\Lambda[(\Lambda - Q)^{-1}\Lambda]^{k+1} - ep(\Lambda - Q)^{-2}\Lambda e. \end{aligned} \tag{2.2}$$

sachant que $p$ est le vecteur stationnaire de $F(\infty) = (\lambda - Q)^{-1}\Lambda$.

## 2.9 Le Modèle

En choisissant un modèle de Poisson. le taux d'erreur serait égal à $E[X]^{-1}$. Nous avons choisi un modèle à deux états décrit par la figure 2.2, dans lequel les taux d'arrivées représentent les taux d'erreur. L'état $i$ à un taux d'erreur de $\lambda_i$.

Le but était d'utiliser les données des tests effectués pour pouvoir estimer les paramètres des équations données par le modèle MMPP. Les quatre paramètres qui sont $\lambda_0, \lambda_1, \sigma_0$ et $\sigma_1$ définissent complètement le processus MMPP.

La distribution des délais inter-pertes est présentée par la figure 2.1. Le logiciel Splus a été utilisé pour essayer de trouver la distribution qui se rapproche le plus de nos données, sans succès; la distribution n'est pas connue. Les moments et les corrélations ont été calculés à partir des tests réalisés. Nous avons obtenu :

$$E[X] = \frac{1}{N}\sum_{j=1}^{N} X = 125.11 \tag{2.3}$$

$$E[X^2] = \frac{1}{N}\sum_{j=1}^{N} X^2 = 97418.1 \tag{2.4}$$

$$E[X^3] = \frac{1}{N}\sum_{j=1}^{N} X^3 = 131133821 \tag{2.5}$$

$$E[(X - E[X])(X - E[X])] = -13441 \tag{2.6}$$

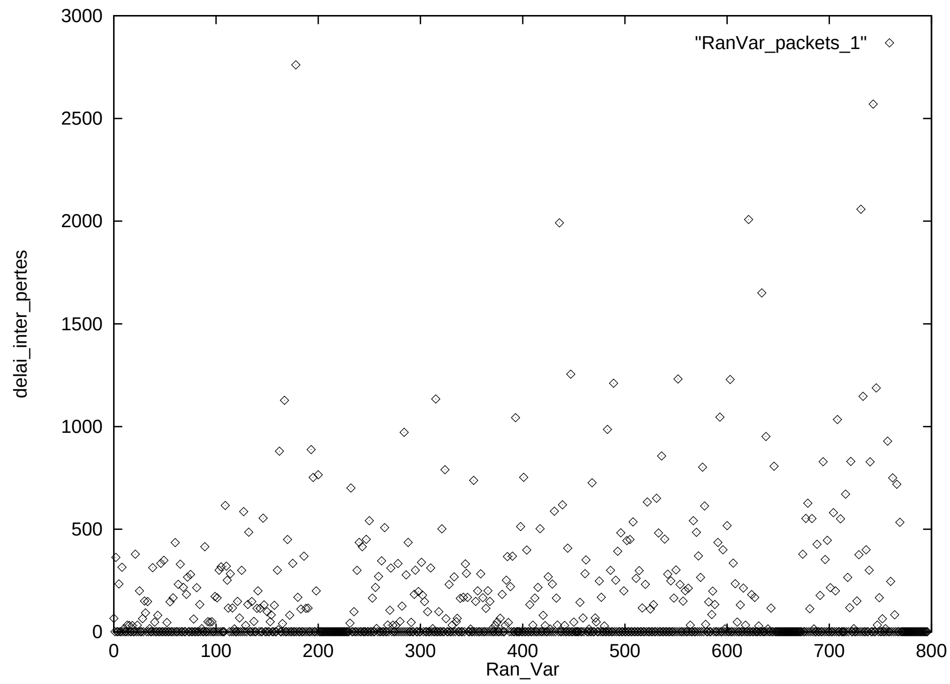


FIG. 2.1: *Distribution des délais inter pertes*

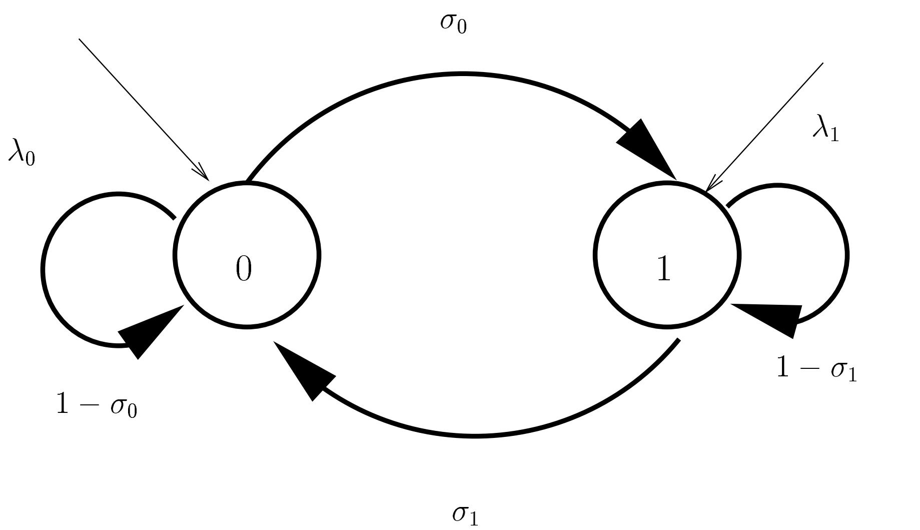


FIG. 2.2: *Modèle MMPP à deux états*

$X$ étant une vriable aléatoire associée aux délais inter-pertes. A l'aide des équations 2.1 et 2.2, nous avons essayé d'identifier $\lambda_0$, $\lambda_1$, $\sigma_0$, $\sigma_1$, sans succès[5]. En utilisant l'éspérence, le $2^{nd}$ moment et en considérant un modèle IPP[6]. Sachant que dans ce dernier cas, on ne peut pas utiliser les équations des corrélations. Nous avons obtenu les solutions du système d'équations qui sont données par :[7]

$$\lambda_2 = 11.4, \sigma_1 = 1.36, \sigma_2 = .49. \tag{2.7}$$

et qui défінssent complètement le processus. Cette modélisation (reduction du modèle MMPP en un modèle IPP) n'étant pas satisfaisante, nous essayons actuellement de le résoudre autrement.

5. Nous avons pour cela utilisé MAPLE.
6. Interrupted Poisson Process i.e. On fixe un des taux d'arrivées à zéro.
7. $\lambda_1 = 0$

# Quatrième partie

# Conclusion

L'étude menée durant ce stage a porté sur deux aspects :

- le contrôle de flux : nous proposons deux nouvelles améliorations au protocole TCP :
  - poser le seuil de congestion égal à $W_{max}\alpha$ pour éviter d'avoir une perte durant la phase "slow start".
  - pendant la phase "slow start", améliorer l'algorithme de l'émetteur en estimant dynamiquement la taille de la bande passante afin d'éviter le doublement automatique de son débit à la réception d'un acquittement.
- modélisation des erreurs du lien satellite. Les paramètres de la FEC ont été modifiés dans le but de modéliser les pertes de paquets. On a constaté qu'il n'y avait pas de pertes de paquets dans des conditions atmosphériques normales. Pour les obtenir on peut procéder de plusieurs manières : en dépointant l'antenne de réception, en attendant les conditions météo favorables (pour les pertes) ou en baissant la puissance d'émission de l'antenne. Nous avons opté pour la troisième possibilité, il est clair qu'en baissant la puissance de l'antenne d'émission, il en résulte à la réception des pertes de données.

  Les tests ont montré que ces pertes étaient liées ("bursty losses"). Le modèle utilisé est basé sur un processus MMPP à deux états. Mis à part dans le cas de conditions météo ,il n'est pas courant d'avoir recours aux deux autres méthodes pour obtenir des pertes : en général, l'antenne pointe vers la même position et sa puissance d'émission est reglée une fois pour toutes. Mais cette modélisation servira par la suite quand même car dans des conditions differentes on peut avoir ces pertes.

Cette étude a été entreprise dans le but d'obtenir une modélisation des erreurs. Celle-ci permettra de définir de "bons" codes correcteurs, qui ne soient pas gourmands en bande passante.

# Cinquième partie

# Annexes

# Chapitre 1

# Annexe 1

On a utilisé trois niveaux différents de FEC pour effectuer nos tests qui sont (3/4, 1/2, 7/8). Le but étant :

- d'étudier des pertes au niveau du paquet,
- d'étudier des pertes au niveau du caractère (bits),
- de générer les traces et de les garder car on ne peut pas modifier le taux de FEC sur un lien satellite à la demande (ces traces pourront servir ensuite pour effectuer des expériences sur des réseaux de simulation comme c'est le cas pour les réseaux mobiles).

Après avoir modifié le taux de FEC, il faut calculer le taux d'erreurs moyen, et à partir de ce taux calculer la taille moyenne de l'échantillon (le nombre de paquets) qu'il faudra construire pour connaître la durée nécessaire des tests. Les paquets envoyés sont munis d'un checksum UDP pour pouvoir identifier le paquet: son numéro de séquence ainsi que sa taille (et bien sûr le checksum ne couvre que l'en-tête) i.e.: on peut recevoir des paquets erronnés: les bits du champ de données sont altérés.

On a une source qui génère des paquets à intervalles réguliers. On teste avec différentes tailles de paquets : 128, 512,1K, 4K octets.

On veut analyser les paquets arrivant et les comparer aux paquets originaux, pour effectuer le maximum de statistiques.

## 1.1 Tests niveau paquets

Le protocole de transport utilisé pour l'envoi des paquets est le protocole UDP, car TCP est un protocole fiable qui se charge des retransmissions et corrections; or ce qu'on veut c'est obtenir des pertes à la destination.

Le protocole UDP a une option udp-cksum qui est mise à un en fonctionnement normal, cela signifie que le calcul du checksum est effectué à la destination. En la positionnant à zéro, on reçoit les paquets erronés (qui auraient dû être rejetés au niveau transport).

La machine réceptrice mesure les délais inter-paquets perdus pour essayer de trouver la distribution de ces pertes. On comptabilise les paquets perdus. Un paquet est perdu:

– s'il est rejeté avant d'arriver au niveau UDP (mauvais checksum IP ou rejeté par la carte),

– s'il est arrivé au niveau UDP mais avec un mauvais checksum UDP; dans ce cas il ne pourra pas être comparé â l'original (si le checksum est mauvais : on ne pourra pas savoir de quel paquet il s'agit),

– le paquet peut avoir un bon checksum UDP mais ne pas avoir la bonne taille (on aura perdu des caractères), ce paquet sera perdu aussi.

## 1.2 Tests niveau bits

On fait de même au niveau caractère (bits): On compare les paquets reçus aux paquets originaux car il n'est pas garanti que tous les bits seront conformes (identiques au paquet original).

Dans cette expérience on s'intéresse à la variable aléatoire $RanVar_p$, qui représente les temps entre les pertes de paquets consécutifs, et à la variable aléatoire $RanVar_c$, qui représente les temps entre les pertes de caractères consécutifs. La granularité est de l'ordre des numéros de séquences. $RanVar_p$ et $RanVar_c$ prennent des valeurs discrètes. I étant l'intervalle dans lequel $RanVar_p$ prend ses valeurs: $1 \ldots$ MAX. MAX étant le numéro de séquence maximum.

Si les variables aléatoires sont indépendantes (i.e. les intervalles entre deux erreurs consécutives sont indépendants) on a le carré de l'espérance qui est égal à l'espérance du produit des variables aléatoires.

La machine émettrice ainsi que la réceptrice enregistrent les temps d'envoi et de réception des paquets pour voir s'il y a beaucoup de variations dans les délais.

# Chapitre 2

# Annexe 2

Un code binaire (noté $C(m,n,p)$) est une partie de $\{0,1\}^n$, de cardinal $M$. Le taux de transmission d'un code $C$ est défini par : $R = \frac{1}{n} log_2 M$ on a $0 \leq R \leq 1$. Le théorème de Shanon nous dit qu'il existe un code associé à un schéma de décodage tel que la probabilité de décodage incorrect du mot reçu soit aussi faible que l'on veut. La condition nécéssaire et suffisante est que le taux de transmission $R$ du code soit inférieur à la capacité du canal. Ce théorème ne nous dit pas comment construire ce code. Le deuxième théorème de Shanon dit que l'on peut transmettre de l'information à travers un canal bruyant à condition d'ajouter suffisamment de redondance.

$GF(p^n)$ est un corps de cardinal $p^n$. Soit $V = [GF(2)]^n$ l'espace vectoriel de dimension $n$ sur $GF(2)$.

Un code binaire $C$ est un sous ensemble de $V$, il est linéaire s'il a une structure de sous espace vectoriel.

Un code linéaire $C$ de longueur $n$, est dit cyclique si toute permutation cyclique d'un mot du code est aussi un mot du code. Les codes $BCH$ sont des codes cycliques.

Les codes de REED-SOLOMON sont des codes $BCH$ à coéfficients dans $GF(p^m)$, dont la longueur est $n = p^m - 1$ et de distance minimum la plus large possible (i.e. $d = n - k + 1$).

La loi de Shanon-Hartley détermine le débit théorique maximum des informations d'un canal de transmission en fonction de la bande passante $W$ et du taux $\frac{S}{N}$ $R_{max} = W log_2(1 + \frac{S}{N})$ bps. Pour un certain taux d'erreurs (BER) on peut déterminer le débit maximum.

# Bibliographie


[All97] Mark Allman. Improving TCP performance over satellite channels. Master's thesis, Ohio University, 1997.

[EA95] Philippe Nain Driss Elouadghiri Erramdani Eitan Altman, Jean Bolot. Performance modeling of TCP/IP in a wide-area network. *IEEE Conf. on Decision and Control*, 1995.

[Fri95] Daniel E. Friedman. *Error Control for Satellite and Hybrid Communication Networks.* PhD thesis, Dept. of Electrical Engineering, 1995.

[GC92] Philippe Goldlewski Gerard Cohen, Jean Louis Dornstetter. *Codes correcteurs d'erreurs.* MASSON, 1992.

[Jac] Van Jacobson. Congestion avoidance and control.

[Jun97] Matthias Jung. *Error Control for Reliable Multicast over Satellite Links.* PhD thesis, Eurecom France, 1997.

[Lak97] T.V. Lakshman. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM*, 1997.

[MA98a] C. Partridge M. Allman, S. Floyd. Increasing TCP's initial window. Technical report, NAS Lewis, 1998.

[MA98b] Dan Glover Mark Allman. Enhancing TCP over satellite channels using standard mechanisms. Technical report, NASA Lewis, 1998.

[MA98c] Shawn Ostermann Mark Allman, Chris Hayes. An evaluation of TCP with larger initial windows. *NASA's Lewis Research Center Satellite Networks and Architectures Branch.*, 1998.

[MAT98] J. Griner K. Scott M. Allman, D. Glover and J. Touch. Ongoing TCP research related to satellites. Technical report, IETF, 1998.

[MF95] Steven McCanne and Sally Floyd. Ns( network simulator). *URL http://www-nrg.ee.lbl.gov/ns*, 1995.

[MMR96] Sally Floyd Matt Mathis, Jamshid Mahdavi and Allyn Romanom. RFC : 2018. TCP selective acknowledgement options. Technical report, 1996.

[nA93] Nihat Cem Oguz nder Ayanoglu, Richard Gitlin. Performance improvement in broadband networks using fec for lost packet recovery. *Journal of high speed Networks*, 1993.

[Par97] C. Partridge. Ack spacing for high delay-bandwidth paths with insufficient buffering. Technical report, End-To-End Research Group, 1997.

[Riz97] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review*, 1997.

[SL84] Michael J. Miller Shu Lin, Daniel Costello. ARQ. *IEEE Communications Magazine*, 1984.

[Ste97] W. Stevens. RFC 2001. Technical report, Request For Comment, 1997.

[WF93] Kathleen Meier-Hellstern Wolfgang Fisher. The Markov-Modulated Poisson process (MMPP) cookbook. *Performance Evaluation*, 1993.