Haute Ecole Spécialisée
de Suisse Occidentale

Jürgen Ehrensberger

Chapitre 4
La couche liaison

1 But du chapitre

Ce chapitre commence par décrire le service que la couche liaison de données offre à la couche supérieure. Ensuite, il examine en détail les fonctions nécessaires afin de réaliser ce service. Ces fonctions sont

- Le découpage en trames
- Le contrôle d'erreurs
- La retransmission de paquets et la temporisation
- L'acquittement de données et l'utilisation des numéros de séquence
- Le contrôle de flux
- L'établissement et la terminaison de connexion.

Les différents algorithmes pour réaliser chacune des fonctions seront discutés. A la fin de ce chapitre, les protocoles les plus importants (HDLC, LAPB, LAP, SLIP, PPP) de la couche liaison sont présentés.

2 Introduction

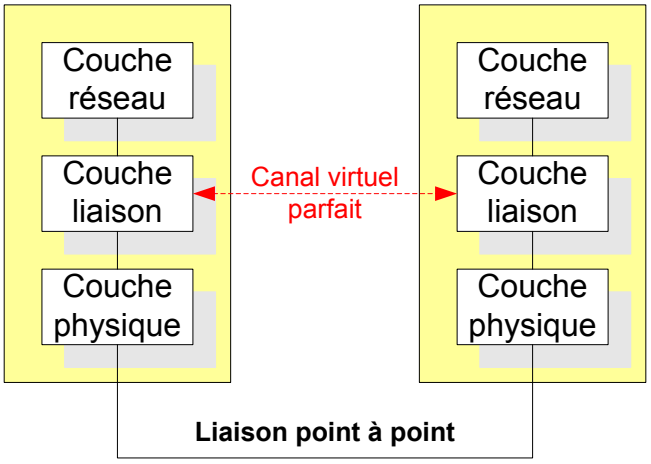
La couche liaison de données et la deuxième couche du modèle de référence OSI. Elle permet à deux stations adjacentes dans un réseau de communiquer de façon fiable et efficace. Par adjacentes, nous entendons que les deux stations sont physiquement connectées par un canal de transmission (voir Chapitres 2 et 3) qui délivre les bits dans l'ordre dans lequel ils ont été émis.

A première vue, le problème peut sembler élémentaire : la station A se contente d'émettre ses données sur le support physique et la station B de les collecter. Malheureusement, les liaisons de transmission ne sont pas parfaites : le débit binaire est limité, le délai de propagation est non nul, des erreurs de transmission peuvent survenir... Ces facteurs influent fortement sur le transfert des données et doivent être pris en compte dans l'élaboration des protocoles.

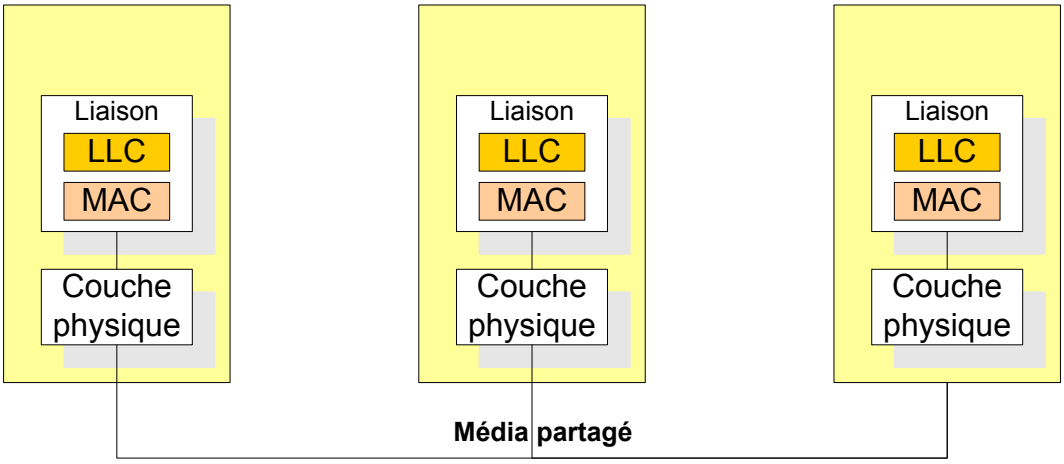
Les protocoles de la couche liaison dans les réseaux WAN sont souvent très différents des protocoles dans les réseaux LAN. Une liaison WAN connecte typiquement deux nœuds d'un réseau (liaison point à point) à travers une distance importante. A cause de la distance, ces liaisons sont susceptibles à des taux d'erreurs relativement importants, surtout si le media de transmission est une ligne électrique ou un canal radio. Afin de réaliser un service fiable, vue par les couches supérieures comme une liaison parfaite, les fonctions de détection et de correction d'erreurs prennent un rôle important.

La situation est différente dans les réseaux locaux. Ici, les distances entres les nœuds du réseau sont courtes et les erreur bits beaucoup plus rares. Cependant, les technologies LAN utilisent souvent un canal partagé comme media de transmission sous-jacent, comme par exemple un bus coaxial dans les premières générations d'Ethernet ou un canal radio dans les réseaux locaux sans fils modernes. La gestion du canal partagé, notamment la gestion de l'accès au canal, demande des mécanismes souvent complexes qui sont très différents des mécanismes utilisés sur les liaisons

point à point. Cette différence se reflète dans les modèles en couches des réseaux WAN et LAN (Figure 1). Dans les réseaux LAN, la couche liaison de données est sous-divisée en une sous-couche LLC (Logical Link Control) et une sous-couche MAC (Medium Access Control). La sous-couche MAC gère l'accès au media partagé. La sous-couche LLC peut réaliser un service fiable, bien que cette fonctionnalité ne soit pas souvent mise en œuvre en pratique.



a) Couche liaison dans un réseau WAN



a) Sous-division de la couche liaison dans un réseau LAN

Figure 1: La couche liaison de données dans les réseaux WAN et LAN.

3 Objectifs de la couche liaison

La couche liaison de données doit réaliser un certain nombre de fonctions spécifiques. Elle offre une interface de service clairement définie à la couche réseau. Elle détermine la façon dont les bits venant de la couche physique sont regroupés en trames et se charge de traiter les erreurs de transmission. Elle effectue enfin un contrôle de flux pour régulariser la vitesse d'échange des données entre entités source et destination.

Vu depuis la couche réseau, donc depuis l'utilisateur du service, la fonction principale de la couche liaison consiste à transmettre les données de la couche réseau d'une station source à la couche réseau d'une station destination. D'après la fiabilité de ce service, on peut le classer en trois catégories :

1. Le service **sans connexion** et **sans accusé de réception**.
2. Le service **sans connexion** et **avec accusé de réception**.
3. Le service **orienté connexion avec accusé de réception**.

Analysons-les brièvement.

Dans le service 1, la station source envoie des trames à la station destination sans recevoir d'accusés de réception de cette dernière. Aucune connexion n'est établie au préalable, ni libérée après l'envoi des données. Si une trame est perdue (à cause de bruit sur la ligne, par exemple), il n'est prévu dans cette couche aucun moyen de remédier à cette perte. Ce type de service convient lorsque le taux d'erreur est faible et que la correction des erreurs de transmission a été prévue dans les couches supérieures. Il est également utilisé dans la plupart des réseaux locaux, ainsi que dans le cas des trafics en temps réel, par exemple la parole numérisée, pour lesquels il vaut mieux avoir des données légèrement altérées que des données en retard.

Le service 2, sans connexion mais avec accusé de réception, est plus fiable que le précédent. Il n'existe toujours pas de connexion mais chaque trame envoyée est acquittée. L'émetteur sait pour chaque trame si elle a été correctement reçue. Si la trame n'est pas arrivée au bout d'un certain temps, il peut l'envoyer de nouveau. Ce service s'avère très utile avec des canaux peu fiables comme le sont les liaisons sans fil.

Il est important de bien préciser que la fourniture d'accusés de réception au niveau de la couche liaison correspond à une optimisation de cette couche et non à une exigence. La *couche transport* peut toujours envoyer un message et attendre son acquittement. Si celui-ci n'est pas arrivé au bout d'un temps déterminé, elle peut toujours réémettre le message. L'inconvénient de cette méthode est que si un message est, en moyenne, divisé en 10 trames et que 20 pour cent des trames sont perdues, la transmission d'un message entier risque de prendre beaucoup de temps. En revanche, si on acquitte individuellement chaque trame (au niveau de la couche liaison), la transmission du message sera beaucoup plus rapide. Si avec des canaux fiables, comme la fibre optique, le surcoût de la gestion de ces acquittements peut être économisé, il n'en est pas de même avec des canaux peu fiables comme les liaisons sans fil.

Le service le plus élaboré est le service 3. Avant tout envoi de données, les stations source et destination doivent établir une connexion. Chaque trame envoyée sur la

connexion est numérotée et la couche liaison garantit que chaque trame envoyée est reçue. Plus précisément, la couche liaison de données garantit que chaque trame est reçue une fois et une seule et que toutes les trames sont reçues dans l'ordre d'émission. En revanche, dans un service sans connexion, il est possible qu'un acquittement perdu provoque la retransmission d'une trame plusieurs fois et, de ce fait, que le récepteur la reçoive plusieurs fois. Un service avec connexion fournit donc aux deux processus de la couche réseau l'équivalent d'un canal fiable.

4 Découpage en trames

Pour être en mesure de fournir un service à la couche réseau, la couche liaison de données doit utiliser le service de la couche physique. Celle-ci assure le transport de trains de bits sur le support et leur remise à la station de destination. Ces trains de bits peuvent comporter des erreurs. Le nombre de bits reçus peut être inférieur, égal ou supérieur au nombre de bits émis ; les bits peuvent également avoir changé de valeur. La couche liaison de données doit détecter ces erreurs et les corriger si nécessaire.

À cette fin, elle découpe généralement le train de bits en trames. Chaque trame est structurée en plusieurs champs qui contiennent les données à transmettre entre les entités de la couche réseau, mais aussi des informations de signalisation entre les entités de la couche liaison, comme par exemple une somme de contrôle.

Le découpage en trames des trains de bits est plus difficile qu'il y paraît à première vue. On peut le réaliser en insérant des silences pour délimiter les trames, à l'instar des espaces entre les mots. Mais les couches physiques garantissent rarement les délais : des bits perdus ou des perturbation pourraient fausser les frontières entre les trames.

C'est pour cette raison que l'on a envisagé d'autres méthodes. Dans cette section, nous en décrirons quatre

1. Compter les caractères.
2. Avoir des caractères de début de trame, de fin de trame et de transparence.
3. Utiliser des fanions de début et de fin de trame avec des bits de transparence.
4. Violer le codage normalement utilisé dans la couche physique.

4.1 Comptage de caractères

La première méthode utilise dans l'en-tête de la trame un champ qui indique le nombre de caractères qu'elle contient. Quand la couche liaison de données de la station destination lit ce champ, elle connaît le nombre de caractères de la trame. Cette technique est illustrée par la Figure 2a) qui montre quatre trames contenant respectivement 5, 5, 8, et 8 caractères.

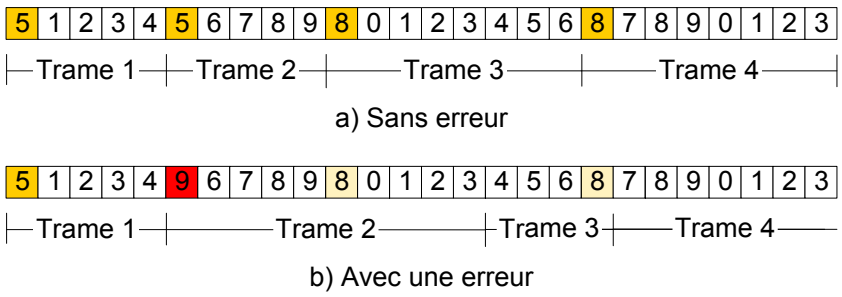


Figure 2: Découpage en trames avec un compteur de caractères

Cet algorithme pose un problème lorsque le caractère indiquant le nombre de caractères dans la trame est affecté par la transmission. Par exemple, si le caractère « nombre de caractères » de la seconde trame (Figure 2b) est changé de 5 en 9, le récepteur perd la synchronisation et ne peut plus trouver le début de la trame suivante.

Grâce à la somme de contrôle, le récepteur se rend compte que cette trame est mauvaise, mais il ne sait toujours pas où commence la suivante. S'il décide de demander à l'émetteur de retransmettre les données, il ne connaît pas le nombre de caractères à laisser passer pour trouver le début de la retransmission. Pour cette raison, cette méthode n'est que rarement utilisée.

4.2 Découpage orienté caractère

La seconde méthode résout le problème de la synchronisation après une erreur de transmission en délimitant chaque trame par les séquences de caractères **DLE STX** et **DLE ETX**. Les séquences DLE STX (*Data Link Escape*, *Start of TeXt*) et DLE ETX (*Data Link Escape*, *End of TeXt*) sont placées respectivement en début et en fin de trame. Lorsque la station destination perd la synchronisation, il lui suffit de rechercher les séquences DLE STX ou DLE ETX pour retrouver la délimitation des trames.

L'inconvénient de cette méthode apparaît lorsque les données à transmettre prennent toutes les configurations binaires possibles. La couche liaison simule un canal transparent à la couche réseau, donc sans aucune restriction quant aux données qui peuvent être transmises.

Il se peut alors que les séquences DLE STX et DLE ETX apparaissent dans les données et trompent le récepteur sur la délimitation de la trame. On résout ce problème en ajoutant à l'émission un caractère DLE (un **DLE de transparence**) devant tout DLE apparaissant dans les données à transmettre. La couche liaison de données de la station destination enlève les caractères DLE ajoutés dans les données par la station source avant de les transmettre à la couche réseau. Cette technique appelée « *Byte stuffing* » permet d'obtenir la transparence des données à transmettre. On distingue ainsi les séquences de délimitation de la trame, DLE STX et DLE ETX, des caractères figurant au sein des données à transmettre en recherchant les caractères DLE simples. La Figure 3 montre une chaîne de caractères à transmettre et la trame complète pour la transmettre.

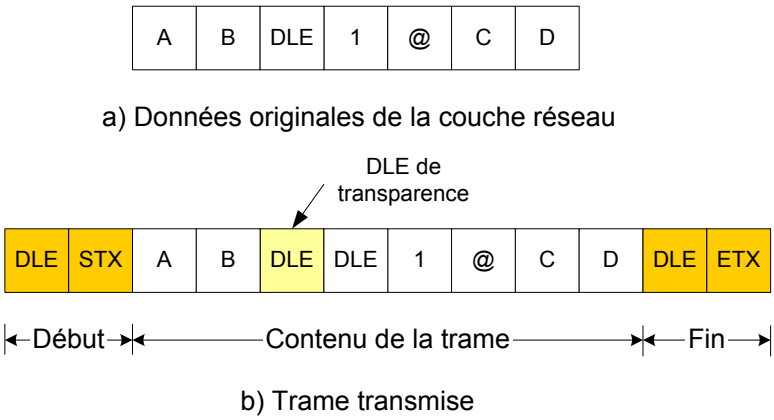


Figure 3: Découpage orienté caractère avec "Byte stuffing"

Regardons ce qui se passe en cas d'erreurs bit. Si une erreur modifie la séquence DLE STX en début de trame, la couche liaison va ignorer cette trame et attendre la prochaine séquence DLE STX. Cette trame est alors perdue et d'autres mécanismes, soit à la couche liaison soit à une couche supérieure, doit se charger de la retransmission si nécessaire. Si la séquence DLE ETX à la fin de la trame est modifiée, la fin de la trame n'est pas correctement détectée. La couche liaison

détectera le début de la trame suivante et jettera la trame précédente comme non valable. Le problème le plus difficile se produit lorsque des bits à l'intérieur de la trame sont modifiés de telle façon que la séquence DLE ETX en résulte. Comme tous les protocoles de la couche liaison utilisent une somme de contrôle, le problème sera très probablement détecté. Les caractéristiques de la somme de contrôle déterminent alors la probabilité que la couche liaison interprète cette trame comme trame valable.

L'inconvénient de cette méthode de délimitation des trames est qu'elle est liée à l'emploi de caractères codés sur 8 bits et plus particulièrement à l'utilisation du codage ASCII. Avec le développement des réseaux, il s'est avéré de plus en plus contraignant d'associer le codage des caractères au mécanisme de délimitation de la trame ; on a donc cherché une nouvelle technique de délimitation qui permette de traiter des caractères d'une taille quelconque. C'est le découpage à l'aide d'un fanion, orienté bit.

4.3 Découpage à l'aide d'un fanion

Cette technique permet

- à la trame de données de contenir un nombre quelconque de bits
- l'utilisation de codages dans lesquels chaque caractère est représenté par un nombre quelconque de bits.

Chaque trame commence et finit par une séquence particulière de bits **01111110** appelée **fanion** (*flag*). Pour rendre le canal transparent pour les données de la couche réseau on utilise une méthode appelée « *bit stuffing* » qui évite que des données soient interprétées comme fanion. Lorsque la couche liaison de données source détecte cinq 1 consécutifs dans les données à transmettre, elle ajoute à leur suite un bit 0 avant d'envoyer le train de bits sur la ligne. Ce **bit de transparence** est analogue au caractère de transparence DLE ajouté aux données à transmettre devant chaque caractère DLE.

Quand le récepteur reçoit cinq bits 1 consécutifs suivis d'un 0, il enlève automatiquement ce dernier. La couche réseau ignore totalement ce mécanisme (c'était déjà le cas avec les caractères de transparence DLE). Si les données de l'utilisateur contiennent le fanion 01111110, la séquence transmise sera 011111010 ; à la réception, on enlève le bit de transparence. La Figure 4 donne un exemple de l'utilisation des bits de transparence.

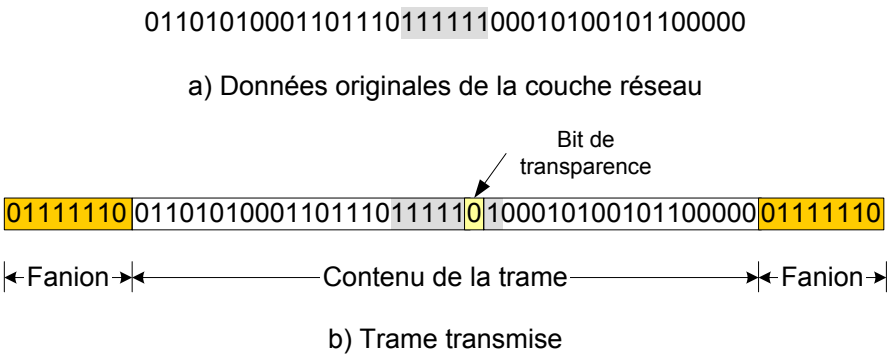


Figure 4: Découpage à l'aide d'un fanion, avec bits de transparence

Grâce aux bits de transparence, le récepteur reconnaît sans équivoque les séquences de délimitation de la trame. Lorsque le récepteur perd la synchronisation, par exemple à cause d'une erreur bit, il lui suffit de rechercher, parmi les trains de bits reçus, le prochain fanion, qui indique avec une très forte probabilité le début de la trame suivante.

4.4 Découpage à l'aide d'une violation du codage physique

La dernière méthode de délimitation de trame est employée lorsque le codage sur le support physique contient des redondances. Ainsi, sur certains réseaux locaux, on code un bit de données en utilisant deux bits physiques : on représente le bit 0 par une impulsion positive suivie d'une impulsion négative et le bit 1 par une impulsion négative suivie d'une impulsion positive (code Manchester). Les combinaisons positive-positive et négative-négative ne sont pas utilisées pour coder des données. Certains protocoles utilisent une séquence invalide telle que deux impulsions positives suivies de deux impulsions négatives pour délimiter la trame. Cette méthode ne nécessite pas l'ajout de bits de transparence dans les données à transmettre.

Pour conclure sur cette délimitation de la trame, ajoutons que certains protocoles utilisent une double méthode : le comptage des caractères et l'une des trois autres méthodes. Lorsqu'une trame arrive, on utilise le compteur de caractères pour rechercher la fin de la trame. Si la séquence de délimitation est repérée et si le contrôle d'erreur est correct, la trame est acceptée. Dans les autres cas, on recherche dans les données reçues la séquence de délimitation suivante.

5 Contrôle d'erreur

Lorsque des données sont transmises à travers un canal physique des perturbations électromagnétiques se superposent au signal analogique. Si le bruit est trop fort il peut arriver que le signal qui représente un bit 0 soit interprété comme un bit 1 par le récepteur, ou inversement. Bien que quelques media, comme les fibres optiques, présente une très bonne immunité contres des perturbations et offre un taux d'erreurs bits très faibles, d'autres media comme les boucles locales utilisées pour ADSL et la canal radio pour les réseaux locaux sans fils peuvent présenter des taux d'erreurs bits très élevés de l'ordre de 10^{-5}. Afin de assurer que les données reçues soient identiques aux données transmises le récepteur doit être en mesure de vérifier que les données soient correctes.

En raison même des processus physiques qui les engendrent, les erreurs qui apparaissent avec certains medias comme le canal radio se produisent par rafales, ce qui a des avantages et des inconvénients par rapport aux erreurs isolées qui ne portent que sur un seul bit. Parmi les avantages on peut mentionner le fait que les données sont toujours envoyées sous forme de blocs de bits. Supposons que la taille des blocs soit de 1000 bits et que le taux d'erreur soit de 1 pour mille. Si les erreurs sont indépendantes, la plupart des blocs risquent de contenir une erreur ; en revanche, si les erreurs se produisent par rafales de 100 ou plus, seuls un ou deux blocs sur 100 seront, en moyenne, affectés. L'inconvénient des erreurs en rafale est qu'elles sont beaucoup plus difficiles à détecter et à corriger que les erreurs isolées.

Les concepteurs de réseau ont développé deux stratégies dans le domaine des erreurs de transmission. La première consiste à inclure dans les blocs de données suffisamment de redondance pour que le récepteur soit capable de restituer les données originales à partir des données reçues. La seconde consiste à ajouter juste assez de redondance dans les données à transmettre pour que le récepteur puisse détecter les erreurs et demander alors la retransmission des trames erronées. La première stratégie utilise des **codes correcteurs d'erreur** et la deuxième des **codes détecteurs d'erreur**.

5.1 Codes simples : les parités

La méthode la plus simple de détection d'erreur et d'ajouter un seul bit, appelé bit de parité, à un bloc de bits de données. Par exemple, le bit de parité peut avoir la valeur 1 si le nombre de bits 1 dans le bloc de données est impair et 0 s'il est pair. Autrement dit, le bit de parité est la somme, modulo 2, des bits du bloc de données. Le nombre total de bits 1 dans le bloc de données plus le bit de parité est alors toujours pair, d'où le nom **parité paire**. Le contrôle de parité avec une parité impaire fonctionne de manière analogue.

Si lors de la transmission de la séquence des bits (y compris la parité) un seul bit est modifié, alors le récepteur détectera l'erreur et supprimera la trame. Le récepteur pourra également détecter toutes les erreurs de 3 bits, 5 bits, …, donc toutes les erreurs d'un nombre impair de bits. Et cela pour une longueur quelconque du bloc de bits. Cette caractéristique est assez remarquable pour une méthode si simple. Le grand défaut du contrôle de parité simple est qu'il ne permet de détecter qu'un nombre d'erreurs impair. Toutes les erreurs sur un nombre pair de bit passent inaperçues.

Et si on combinait ce contrôle de parité avec une autre méthode aussi simple pour détecter toutes les erreurs sur un nombre pair de bits ? Hélas, une telle méthode n'existe pas et – comme nous allons le voir dans la suite – ne peut pas exister.

Malgré sa simplicité, un simple bit de parité est inadéquat pour la plupart de systèmes de transmission. En pratique, il ne détecte souvent que la moitié des blocs erronés. Il y a deux raisons pour ce comportement faible. La première est que la plupart des modems transmettent plusieurs bits par symbole élémentaire (plusieurs bits par moments) que qu'une erreur d'un symbole élémentaire typiquement entraîne plusieurs bits erronés. La deuxième raison est que beaucoup de sources de perturbations, comme des décharges électriques ou l'interruption temporaire d'une liaison, provoquent de longues rafales de bits erronés. Pour ces raisons, si plusieurs erreurs apparaissent dans un bloc de bits, un nombre pair d'erreurs bits et aussi probable qu'un nombre impair et un simple bit de parité est donc inapproprié.

5.2 Parités verticales et horizontales

Une autre méthode qui est simple et intuitive pour détecter des erreurs est d'arranger la séquence de bits en une matrice avec un bit de parité pour chaque ligne et un autre bit de parité par colonne (Figure 5).

1	0	0	1	0	1	0	1
0	1	1	1	0	1	0	0
1	1	1	0	0	0	1	0
1	0	0	0	1	1	1	0
0	0	1	1	0	0	1	1
1	0	1	1	1	1	1	0

Parités horizontales

Parités verticales

Figure 5: Parités verticales et horizontales

Le bit de parité dans le coin en bas à droite peut être utilisé comme parité pour les autres bits de parité. Les erreurs qui n'affectent que des bits d'une seule ligne de la matrice peuvent être détectées grâce aux bits de parités des colonnes. De manière similaire, les erreurs qui n'affectent qu'une colonne sont détectées grâce aux parités horizontales. Le défaut de cette méthode est que déjà 4 erreurs bit peuvent faire croire au protocole de liaison que les données sont correctes, comme montré à la Figure 6.

1	0	0	1	0	1	0	1
0	1	1	1	0	1	0	0
1	1	**0**	0	**1**	0	1	0
1	0	0	0	1	1	1	0
0	0	**0**	1	**1**	0	1	1
1	0	1	1	1	1	1	0

Parités horizontales

Parités verticales

Figure 6: Erreur non détectable par les parités verticales et horizontales

5.3 Codes correcteurs d'erreurs

Afin de comprendre comment on traite les erreurs, il est nécessaire de comprendre ce qu'est exactement une erreur. Une trame est formée de *m* bits de données et de *r* bits de contrôle. Si l'on note *n* la longueur de la trame ($n = m + r$), l'ensemble des *n* bits sera appelé mot de code.

Etant donné deux mots de code, par exemple 10001001 et 10110001, il est possible de déterminer de combien de bits ils diffèrent. Dans notre exemple, ils diffèrent de trois bits. Pour le trouver, il suffit d'effectuer un OU EXCLUSIF entre les deux mots de code et de compter le nombre de 1 du résultat. Le nombre de bits de différence entre deux mots de code est appelé **distance de Hamming**. Si la distance de Hamming entre deux mots de code est *d*, il faut *d* erreurs simples pour transformer un mot en l'autre.

Dans beaucoup d'applications, il est possible d'utiliser les 2^m combinaisons des bits de données mais seule une partie des 2^n combinaisons des mots de codes est autorisée. Cela est dû à la façon dont sont calculés les bits de contrôle. Connaissant l'algorithme qui permet de calculer les bits de contrôle, il est possible d'obtenir la liste de tous les mots de code afin de trouver la distance minimale entre deux mots de code. Cette distance est la **distance de Hamming du code** complet.

La propriété d'un code correcteur ou détecteur dépend de sa distance de Hamming.

Pour détecter *d* erreurs, il faut que le code ait une distance de Hamming de *d+1*.

En effet, dans un tel code, il est impossible que *d* erreurs simples changent un mot de code en un autre mot de code autorisé. Lorsque le récepteur reçoit un mot de code non autorisé, il sait qu'une erreur de transmission a eu lieu.

Si l'on veut maintenant être en mesure de corriger *d* erreurs, on doit utiliser un code de distance *2d+1*.

Dans ce cas, la distance entre chaque mot de code est telle que même si *d* erreurs simples se produisent, le mot de code original reste le plus proche du mot transmis: on peut donc le retrouver.

Le code de contrôle de parité simple constitue un exemple simple de code détecteur. On ajoute aux bits de données un bit de parité. Ce code a une distance de 2 puisque toute erreur simple produit un mot de code non autorisé. Il est utilisé pour détecter les erreurs simples.

L'exemple suivant illustre un code correcteur très simple. Considérons un code comprenant uniquement quatre mots de code:

0000000000, 0000011111, 1111100000 et 1111111111

Ce code a une distance de 5 et peut donc corriger les erreurs doubles. Si le récepteur reçoit le mot de code 0000000 111, il sait que le mot original était 00000 11111. Mais si une erreur triple change 0000000000 en 0000000111, le récepteur n'est pas capable de la corriger.

Supposons que nous voulions construire un code qui permette de corriger toutes les erreurs simples, avec *m* bits de données et *r* bits de contrôle. Pour chacun des 2^m

messages possibles, il existe *n* mots de code non autorisés situés à une distance d'une unité du mot de code autorisé. Ils sont obtenus à partir du mot de code autorisé en inversant un de ses *n* bits. À chacune des 2^m combinaisons de bits de données possibles, on associe *n*+1 mots de *n* bits (les *n* mots de code non autorisés définis ci-dessus plus le mot de code autorisé). Comme le nombre total de mots de *n* bits est 2^n, la relation suivante doit être vérifiée : $(n+1)2^m \leq 2^n$. En utilisant l'égalité $n = m + r$, l'inégalité devient

$$(m + r + 1) \leq 2^r .$$

Cette inégalité donne une limite théorique de l'efficacité d'un code correcteur. Il n'est pas possible de construire des codes correcteurs pour des erreurs simples avec moins de bits de contrôle que indiqué par cette inégalité. Connaissant *m*, cette inégalité permet de déterminer le nombre minimal de bits de contrôle nécessaires pour que toutes les erreurs simples soient corrigées.

Hamming décrit une méthode permettant d'atteindre cette limite théorique. L'idée est la suivante :

Les bits des mots de code sont numérotés consécutivement en commençant par celui de gauche qui sera le bit 1. Les bits dont les numéros sont des puissances de 2 (1, 2, 4, 8, 16, etc.) sont des bits de contrôle. Les autres bits (3, 5, 6, 7, 9, etc.) sont des bits de données. Chaque bit de contrôle est choisi de façon qu'une série de bits (lui-même compris) ait une parité paire (ou impaire). Un bit de données peut servir dans le calcul de plusieurs bits de contrôle. Pour connaître les bits de contrôle qui utilisent le bit de données de numéro *k*, il suffit d'écrire k comme la somme de puissances de 2. Par exemple, 11 = 1 + 2 + 8 et 29 = 1 + 4 + 8 + 16. On vérifie un bit de données par les bits de contrôle dont les numéros sont les nombres qui apparaissent dans sa décomposition en puissances de 2 (par exemple le bit 11 est vérifié par les bits 1, 2 et 8).

Quand un mot de code arrive, le récepteur initialise un compteur à zéro. Il examine alors chaque bit de contrôle k (k = 1, 2, 4, 8 ...) pour vérifier s'il a une parité correcte. Si ce n'est pas le cas, il ajoute k au compteur. Lorsque tous les bits de contrôle ont été vérifiés et si le compteur est nul c'est-à-dire si tous les bits de contrôle sont corrects), le mot de code est considéré comme autorisé. Si le compteur est non nul, il contient le numéro du bit erroné. Par exemple, si les bits de contrôle 1, 2 et 8 ne vérifient pas la parité, le bit 11 est erroné car c'est le seul qui contribue au calcul à la fois des bits 1, 2 et 8.

Ce calcul peut facilement être expliqué sous forme matricielle. Les *n* bits d'un mot de code y^T sont constitués par les *m* bits x^T de données suivis d'un vecteur a^T des *r* bits de contrôle calculés par une combinaison linéaire des bits d'information. On a donc

$$y^T = x^T H$$

où *H* est une matrice de dimension $m \times n$, appelé **matrice génératrice du code**, qui se décompose en une matrice unité I_m de dimension $m \times m$ et une matrice *P* de dimension $m \times r$:

$$H = [I_m, P].$$

Regardons un exemple. Pour détecter toutes les erreurs simples sur 4 bits, au minimum 3 bits de contrôle sont nécessaires ($4+3+1 \leq 2^3$). La matrice génératrice H a alors la forme

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 1 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 1 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 0 & 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}.$$

Dans ce cas, les trois bits de contrôle a_1, a_2, a_3, sont définis par

$$a_1 = x_1 + x_2 + x_3$$
$$a_2 = x_2 + x_3 + x_4$$
$$a_3 = x_1 + x_2 + x_4$$

où x_i est le bit de données numéro i. Le bit de contrôle a_1 vérifie alors les bits de données numéro 1, 2 et 3. Le mot de code pour 1010 est alors calculé comme 1010011.

Pour décoder les mots de code, on calcule

$$s^T = y^T G^T,$$

où

$$G^T = \begin{bmatrix} P \\ I_r \end{bmatrix}$$

est la matrice de contrôle de parité.

Si le vecteur s vaut 0, le mot de code est correct, comme

$$y^T G^T = x^T H G^T = x^T \begin{bmatrix} I_m & P \end{bmatrix} \cdot \begin{bmatrix} P \\ I_r \end{bmatrix} = 0 .$$

Dans notre exemple,

$$G^T = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Le test pour le mot de code 1010011 donne

$$(1010011)\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (000).$$

Les codes de Hamming ne peuvent corriger que les erreurs simples. Toutefois, il existe un moyen d'utiliser les codes de Hamming pour corriger des erreurs en rafale. Une séquence de *k* mots de code consécutifs est vue comme une matrice dont les k lignes sont des mots de code. Normalement, les mots de code sont transmis un par un en commençant par le bit le plus à gauche du mot. Pour corriger les erreurs en rafale, les données sont transmises colonne par colonne en commençant par la colonne la plus à gauche. Quand les *k* bits ont été envoyés, la seconde colonne est transmise et ainsi de suite. Lorsque la trame arrive, la matrice est reconstruite colonne par colonne. Si une rafale d'erreurs de longueur *k* altère la trame, il y a au plus un bit erroné dans chacun des mots de code. Comme le code de Hamming peut corriger une erreur par mot de code, on peut corriger le bloc entier de données. Cette méthode utilise *kr* bits de contrôle pour corriger une seule rafale d'erreurs d'au plus *k* bits sur un bloc de *km* bits de données.

5.3.1 Efficacité des codes correcteurs

Le critère principal pour juger l'efficacité d'un code correcteur est le nombre de bits de contrôle qu'il faut ajouter pour corriger toutes les erreurs simples de mots d'une longueur fixe. Un code qui permet d'atteindre la limite théorique, c'est-à-dire pour lequel l'équation $(m+r+1) = 2^r$ est vraie, est dit un code parfait. Un code Hamming avec 4 bits de contrôle permet de corriger toutes les erreurs simples sur 5 à 11 bits de données et il est parfait pour 11 bits de données. Or, si l'on considère des mots de données de 8 bits, ce code implique une charge supplémentaire de 50%. Pour augmenter l'efficacité du code, la seule solution est de considérer des mots de données plus longs. En effet, le **rendement** *R* d'un code défini par $R = m/(m+r)$ donne

$$R = \frac{n-r}{n} = 1 - \frac{\log_2(n+1)}{n}$$

pour un code parfait. *R* tend vers 1 lorsque *n* tend vers l'infini. En pratique, il n'est pas possible d'utiliser des mots de code trop longs comme la probabilité d'erreurs multiples devient plus importante avec la longueur des mots. Pour une transmission efficace, il faut donc chercher d'autres méthodes de correction d'erreurs.

5.4 Codes détecteurs d'erreurs

Les codes correcteurs sont utilisés pour les transmissions de données dans certains cas particuliers, par exemple lorsque le canal est unidirectionnel (simplex) et qu'il est impossible de demander une retransmission. Mais le plus souvent, pour des raisons

d'efficacité et surtout sur les canaux ayant un faible taux d'erreurs bit, on utilise une technique de détection avec retransmission.

Prenons comme exemple un canal avec des erreurs simples et un taux d'erreurs bit BER de 10^{-6}. Considérons des blocs de données de 1000 bits. Si l'on désire construire un code correcteur pour des blocs de 1000 bits, il faut ajouter 10 bits de contrôle ; pour un mégabit de données, on aura 10 000 bits de contrôle. Pour détecter un bloc qui comporte un seul bit erroné, il suffit d'avoir un bit de parité. En moyenne tous les 1000 blocs une erreur se produit et on retransmet un bloc supplémentaire de 1001 bits. Il faut donc 2001 bits supplémentaires (pour détection et retransmission) par mégabit de données, alors qu'il en faut 10 000 avec un code de Hamming.

Bien que une parité simple puisse être employée, on se tourne dans la pratique vers d'autres méthodes plus performantes en utilisant un **code polynomial**, et une variante appelé code **CRC** (code de redondance cyclique). Ces codes présentent l'avantage de s'accommoder d'une longueur variable des mots de données et de conduire à une réalisation très simple du codeur ou du décodeur tout en ayant une bonne capacité de détection de rafales d'erreurs.

Dans les codes polynomiaux, on considère que les bits d'une chaîne de caractères sont les coefficients d'un polynôme. Ces coefficients ne prennent que deux valeurs : 0 ou 1. Un bloc de *k* bits est vu comme la série des coefficients d'un polynôme comprenant *k* termes allant de x^{k-1} à x^0. Un tel polynôme est dit de degré *k*-1. Le bit le plus à gauche est le coefficient de x^{k-1} ; son voisin est le coefficient de x^{k-2}, etc. Par exemple, la chaîne 110001 comprend 6 bits ; elle est représentée par le polynôme $x^5+x^4+x^0$.

L'arithmétique polynomiale est faite modulo 2 et suit les règles de la théorie algébrique. Il n'y a pas de retenue dans l'addition ni dans la soustraction. Faire une addition ou une soustraction équivaut à effectuer un OU EXCLUSIF entre les opérandes. Par exemple

10011011	00110011	11110000	01010101
+11001010	+11001101	-10100110	-10101111
01010001	11111110	01010110	11111010

Les divisions sont effectuées de la même manière qu'en binaire, excepté que la sous, traction est faite modulo 2.

Pour utiliser un code polynomial, l'émetteur et le récepteur doivent se mettre d'accord sur le choix d'un **polynôme générateur** *G(x)*. Le générateur doit avoir son bit de poids fort et son bit de poids faible égaux à 1. Pour calculer la somme de contrôle (*checksum*) d'un bloc de *m* bits (correspondant au polynôme *M(x)*) il faut que le bloc soit plus long que le polynôme générateur. Le principe consiste à coller, à la fin du bloc, des bits de contrôle de façon que la trame (bloc et bits de contrôle) soit divisible par *G(x)*. Quand le récepteur reçoit la trame, il la divise par *G(x)*. Si le reste obtenu est non nul, c'est qu'il y a eu une erreur de transmission.

L'algorithme de calcul des bits de contrôle est le suivant:

1. Soit *r* le degré de *G(x)*. Ajouter *r* zéros après le bit de poids faible du bloc. Il contient ainsi *m* + *r* bits, correspondant au polynôme $x^r M(x)$.
2. Effectuer la division modulo 2 du polynôme $x^r M(x)$ par *G(x)*.

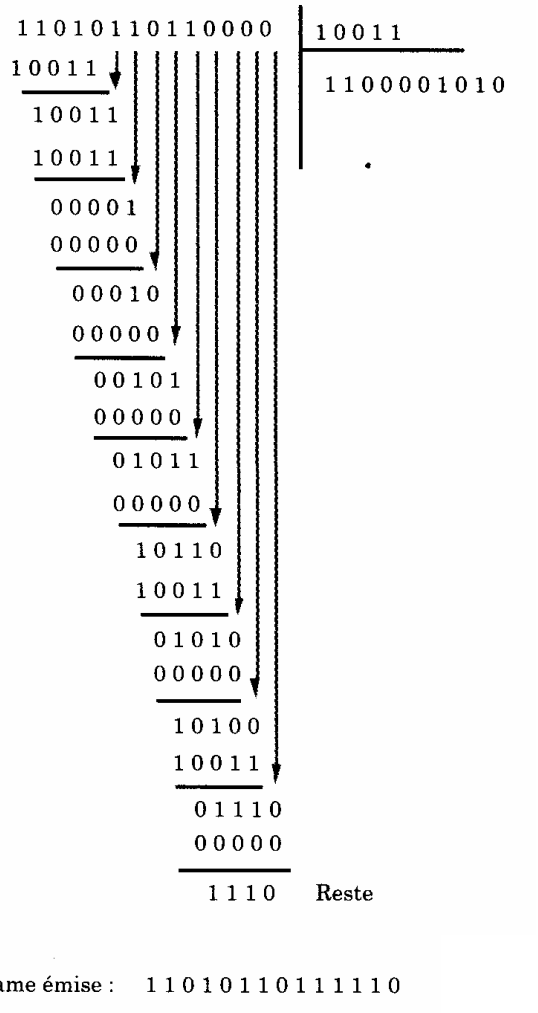
3. Soustraire modulo 2 le reste de la division (qui comprend au plus *r* bits) de la chaîne de bits correspondant au polynôme $x^r M(x)$. Le résultat de cette opération est la trame transmise au destinataire. Appelons *T(x)* le polynôme correspondant.

Les codes polynomiaux peuvent être définis de la façon suivante : **un code polynomial est un code dans lequel tous les mots de code, représentés par des polynômes *T(x)*, sont des multiples d'un polynôme générateur *G(x)*.**

La Figure 7 montre le calcul de la somme de contrôle pour la trame 1101011011 et $G(x) = x^4 + x + 1$.

Trame: 1101011011
Générateur : 10011

Ajout de 4 bits à 0: 11010110110000



Trame émise : 11010110111110

Figure 7: Calcul de la somme de contrôle dans un code polynomial

On doit noter que *T(x)* est divisible (modulo 2) par *G(x)*. En effet, quelle que soit la division effectuée, le dividende moins le reste de la division est toujours divisible par le diviseur. Ainsi, en base 10, si l'on divise 210 278 par 10 941, il reste 2 399. En soustrayant 2 399 de 210 178, on obtient 207 879, nombre divisible par 10 941.

Analysons maintenant l'efficacité de cette méthode. Quelles sortes d'erreurs peut-elle détecter ? Imaginons qu'une erreur de transmission survienne et qu'au lieu de recevoir *T(x)*, le récepteur reçoive *T(x) + E(x)*. Chaque coefficient à 1 dans le polynôme *E(x)* correspond à un bit inversé dans la chaîne transmise. S'il y a *k* coefficients égaux à 1 dans *E(x)*, c'est qu'il s'est produit *k* erreurs simples. Une rafale d'erreurs est caractérisée par un 1, suivi d'un certain nombre de 0 et de 1, puis par un 1 et des zéros.

Lorsqu'il reçoit la trame, le récepteur la divise par *G(x)* ; il calcule *(T(x) + E(x))/ G(x)*. *T(x)/G(x)* étant nul, le résultat de la division est donc *E(x)/G(x)*. Les erreurs qui se traduisent par des polynômes facteurs de *G(x)* ne sont pas détectées. En revanche, toutes les autres le sont.

Si une erreur simple se produit, *E(x)* est égal à x^i où *i* est la place du bit erroné. Si *G(x)* contient au moins deux termes, *E(x)* n'est pas divisible par *G(x)*. Alors toutes les erreurs simples sont détectées.

S'il se produit deux erreurs simples isolées, $E(x) = x^i + x^j$ avec $i > j$, ce que nous pouvons également écrire sous la forme $E(x)=x^j (x^{i-j} + 1)$. Supposons que *G(x)* n'est pas un diviseur de x^j, ce qui est toujours le cas si *G(x)* contient au moins deux termes. Une condition suffisante pour que toutes les erreurs doubles soient détectées est que G(x) ne soit pas un diviseur de $x^k + 1$, pour tout *k* inférieur à la valeur maximale de *i-j* (c'est-à-dire la longueur maximale de la trame). On connaît des polynômes de degré peu élevé qui peuvent être utilisés dans le cas de trames longues. Par exemple, $x^{15} + x^{14} + 1$ n'est pas un diviseur de $x^k + 1$ pour aucun *k* inférieur à 32 768.

Si la trame contient un nombre impair de bits erronés, *E(x)* contient un nombre impair de termes (par exemple $x^5 + x^2 + 1$). Il est intéressant de constater qu'il n'existe pas de polynômes ayant un nombre impair de termes qui possèdent $x + 1$ comme facteur (modulo 2). En construisant un polynôme *G(x)* possédant $x + 1$ comme facteur, on peut détecter toutes les erreurs qui se traduisent par un nombre impair de bits inversés.

Pour démontrer qu'il n'existe aucun polynôme ayant un nombre impair de termes divisibles par $x + 1$, raisonnons par l'absurde et supposons que *E(x)* ait un nombre impair de termes et soit divisible par $x + 1$. *E(x)* peut s'écrire $(x + 1)\ Q(x)$. Calculons ensuite $E(1) = (1 + 1)Q(1)$. Comme $1 + 1 = 0$ (modulo 2), $E(1)$ doit être nul. Or, si $E(x)$ a un nombre impair de termes, la substitution de x par 1 donnera toujours un résultat égal à 1. Il n'existe donc aucun polynôme ayant un nombre impair de termes qui soit divisible par $x + 1$.

Pour terminer, énonçons le résultat suivant: un code polynomial avec *r* bits de contrôle détecte toutes les rafales d'erreurs de longueur inférieure ou égale à *r*. Le polynôme $x^i (x^{k-1} + ... + 1)$ (où *i* détermine à partir de quel bit commence la rafale d'erreurs) représente une rafale d'erreurs de longueur k. Si G(x) contient le terme x^0, il ne peut pas être un diviseur de x^i. Si le degré de l'expression $(x^{k-1} + ... +1)$ est inférieur au degré de *G(x)*, le reste sera donc jamais nul.

Si la longueur d'une rafale d'erreurs est $r + 1$, on peut la représenter par le polynôme $x^i (x^r + ... + 1) = x^i F(x)$.G(x) ne divise F(x) que si les deux polynômes sont identiques, ce qui suppose que les r-1 coefficients autres que le premier et le dernier soient les mêmes pour les deux polynômes. Si toutes les combinaisons de coefficients sont équiprobables, la probabilité pour que les deux polynômes soient identiques est égale à $1/2^{r-1}$. On voit donc que la probabilité de détection d'une salve de longueur r+1 est égale à $1-(1/2^{r-1})$.

En effet, on peut montrer également que la probabilité qu'une trame erronée ne soit pas détectée, lorsqu'une rafale d'erreurs de longueur supérieure à $r + 1$ ou plusieurs courtes rafales d'erreurs surviennent, est de $1/2^{r-1}$ si toutes les configurations binaires sont équiprobables.

5.4.1 Efficacité des codes polynomiaux

L'utilité d'un code détecteur pour des systèmes réels peut être jugé à l'aide de trois critères :

1. La capacité de détecter des erreurs multiples (donc la distance de Hamming minimale du code)
2. La capacité de détection de rafales d'erreurs d'une longueur donnée
3. La probabilité que des erreurs ne soient pas détectées.

En prenant ces critères, les codes polynomiaux s'avèrent bien adaptées aux systèmes de transmission réelle :

1. Toute erreur simple est détectée si le polynôme générateur G(x) comporte plus d'un coefficient non nul.
2. Les erreurs doubles sont toutes détectées si le polynôme générateur G(x) ne divise pas $x^k + 1$, où k peut prendre n'importe quelle valeur comprise entre 1 et n-1.
3. Une erreur sur un message comportant un nombre impair d'erreurs est toujours détectée si le polynôme générateur comporte $(x + 1)$ en facteur.
4. Le code polynomial détecte toutes les rafales d'erreurs de longueur inférieure ou égale au degré r du polynôme générateur G(x).
5. La probabilité de détecter des rafales d'erreurs d'une longueur supérieure à r est de $1-(1/2^{r-1})$.

Pour ces raisons, les codes polynomiaux sont très répandus comme contrôle d'erreur dans les protocoles de liaison de données modernes.

5.4.2 Polynômes générateurs normalisés

Les polynômes générateurs qui ont été normalisés sont

$\text{CRC-12} = x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$

$\text{CRC-16} = x^{16} + x^{15} + x^2 + 1$

$\text{CRC-CCITT} = x^{16} + x^{12} + x^5 + 1$

$\text{CRC-32} = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$

Tous ces polynômes sauf le dernier contiennent $x + 1$ comme facteur premier. Le polynôme CRC-12 est utilisé pour des caractères codés sur 6 bits. Les deux autres sont utilisés pour des caractères codés sur 8 bits. Les polynômes CRC-16 et CRC-CCITT donnent des champs de contrôle d'erreur codés sur 16 bits.

Le polynôme CRC-32 est utilisé comme séquence de contrôle dans Ethernet. Il détecte toutes les rafales d’erreurs de 32 bits. La probabilité de détecter une rafale d’erreurs plus longue est de 99,99999995 %.

Bien qu'à première vue, les calculs nécessaires pour obtenir le champ de contrôle puissent sembler compliqués, mais un simple registre à décalage suffit pour obtenir la somme de contrôle. En pratique, on utilise presque toujours des circuits électroniques pour le codage et le décodage polynomiaux.

Pendant des dizaines d'années on a supposé que les trames à vérifier comportaient des bits dont les valeurs étaient aléatoirement réparties et toutes les recherches d'algorithmes de détection des erreurs ont fait cette hypothèse. On sait maintenant que celle-ci est à revoir et que, dans certains cas, les erreurs non détectées sont beaucoup plus nombreuses qu'on aurait pu le croire.

6 Stratégies de retransmission

Nous avons déjà vu ci-dessus que les codes correcteurs ne s'avèrent pas efficaces pour des liaisons avec un faible taux d'erreurs. La redondance à ajouter aux données pour permettre la correction des erreurs et souvent beaucoup plus importante que la simple retransmission d'un trame erronée, combinée avec un code détecteur. Ces méthodes de retransmission sont généralement appelées **stratégies ARQ** (*Automatic Repeat reQuest*). Dans cette section nous allons étudier les différentes stratégies de retransmission.

6.1 Envoyer et attendre (Stop-and-Go)

Le protocole de retransmission le plus simple est « Envoyer et attendre » ou *Stop-and-Go* en anglais. L'idée principale est de s'assurer que chaque paquet a été reçu correctement avant d'envoyer le prochain. Si une station A transmet des paquets à une station B, elle envoie le premier paquet et attend. Le paquet contient une séquence de contrôle (par exemple un CRC) qui permet au récepteur B de détecter des erreurs bits. Si le paquet est correct, B renvoie un **acquittement** (ACK pour *acknowledgement* en anglais). Si le paquet contient des erreurs, B renvoie un **acquittement négatif** (NAK). Comme des erreurs bit peuvent se produire dans les deux sens de la transmission, les paquets ACK et NAK sont également protégés par une séquence de contrôle.

Lorsque A reçoit un acquittement elle peut envoyer le paquet suivant. Alternativement, si A reçoit un acquittement négatif ou un paquet erroné, elle retransmet le paquet précédent vers B.

Jusqu'ici nous n'avons considéré que le cas où les erreurs de transmission modifient le contenu du paquet. En pratique, la ligne peut subir des coupures temporaires ou présenter un nombre d'erreurs suffisamment élevé pour que les paquets ne soient plus reconnus comme tels. Pour prendre en compte le cas de paquets perdus, la méthode généralement utilisée consiste à enclencher au moment de l'émission de chaque paquet un temporisateur, dont le temps de garde est supérieur au temps aller-retour d'un paquet et son acquittement. Si la station émettrice A ne reçoit pas de d'acquittement avant l'échéance du temporisateur, elle retransmet le paquet précédent.

6.1.1 Numéros de séquence

La méthode décrite ci-dessus fonctionne correctement lorsque le paquet de données a été perdu, mais se trouve en défaut lorsque c'est l'acquittement qui a été perdu. En effet, dans ce dernier cas, la station B a reçu correctement le paquet m_i. Elle le passe donc à la couche supérieure et attend le paquet m_{i+1}.Comme la station A ne reçoit pas d'acquittement avant l'échéance du temporisateur, elle en déduit que le paquet m_i a été perdu et le retransmet, ce qui amène la station B à envoyer deux fois de suite m_i à la couche supérieure. Paradoxalement, les pertes de paquets peuvent donc se traduire par la duplication de certains paquets envoyés à la couche supérieure. Ce type de défaut est illustré à la Figure 8.

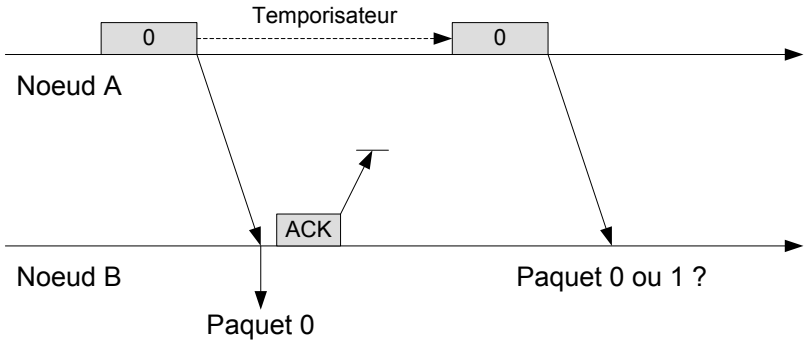


Figure 8: Problème des paquets non numérotés

Pour surmonter les difficultés causées par les retransmissions liées aux pertes de paquets, la solution la plus simple consiste à numéroter les paquets transmis avec un numéro de séquence. Il ne suffit pas de numéroter uniquement les trames de données comme le récepteur doit aussi être capable d'indiquer à quel paquet un acquittement fait référence. Au lieu de renvoyer des paquets ACK ou NAK, le récepteur renvoie un acquittement qui contient le prochain numéro de séquence qu'il attend. Cet ACK numéroté fournit toute l'information des ACK/NAK simples mais évite l'ambiguïté du paquet acquitté. Au lieu du prochain numéro de séquence attendu le récepteur pourrait également indiquer dans son acquittement le dernier numéro de séquence correctement reçu, mais cette convention n'est pas usuelle.

L'acquittement explicit d'un numéro de séquence permet alors l'emploi d'un seul type d'acquittement au lieu de ACK et NAK. Comme le numéro de séquence doit être transmis dans le paquets et les acquittement, le protocole doit prévoir un champ dans l'entête des paquets avec une longueur fixe. Dans la mesure où l'on doit construire un petit entête pour chaque trame, il convient de poser la question suivante : quel est le nombre minimal de bits nécessaire pour coder le numéro de séquence ? Quand la station B reçoit un paquet, il s'agit soit d'une retransmission du paquet m_i soit de la premier transmission du son successeur m_{i+1}. De manière similaire, lorsque la station émettrice reçoit un acquittement, celui-ci accuse la réception soit du dernier paquet transmis, soit de son prédécesseur direct, comme le montre la Figure 9.

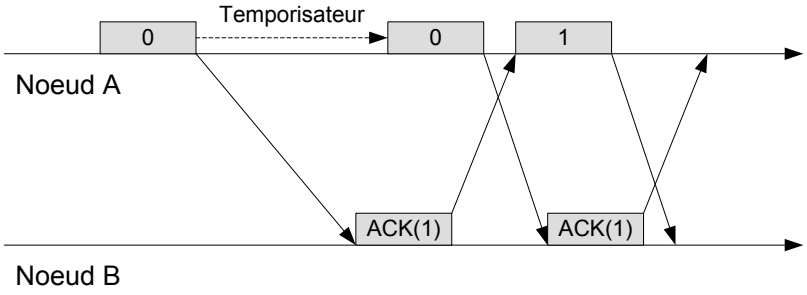


Figure 9: Un acquittement accuse la réception correcte soit du dernier paquet transmis soit de son prédécesseur direct

De ce fait, un numéro de séquence codé sur un bit (0 ou 1) est suffisant. A chaque instant, le récepteur attend un certain numéro de séquence. Lorsqu'un paquet arrive avec un mauvais numéro, il le rejette en considérant qu'il s'agit d'un paquet dupliqué. Lorsqu'un paquet arrive avec le bon numéro de séquence, le récepteur l'accepte et la

transmet à la couche réseau. Le numéro de séquence attendu est incrémenté de 1 modulo 2 et un acquittement est envoyé à l'émetteur.

6.1.2 Algorithme « envoyer et attendre »

En résumé, la Figure 10 montre les algorithmes complets effectués par l'émetteur et par le récepteur pour le protocole « envoyer et attendre ». Afin de vérifier les numéros de séquence, l'émetteur utilise un compteur SN qui contient le dernier numéro de séquence envoyé. De manière analogue, le compteur RN du récepteur contient le prochain numéro de séquence attendu.

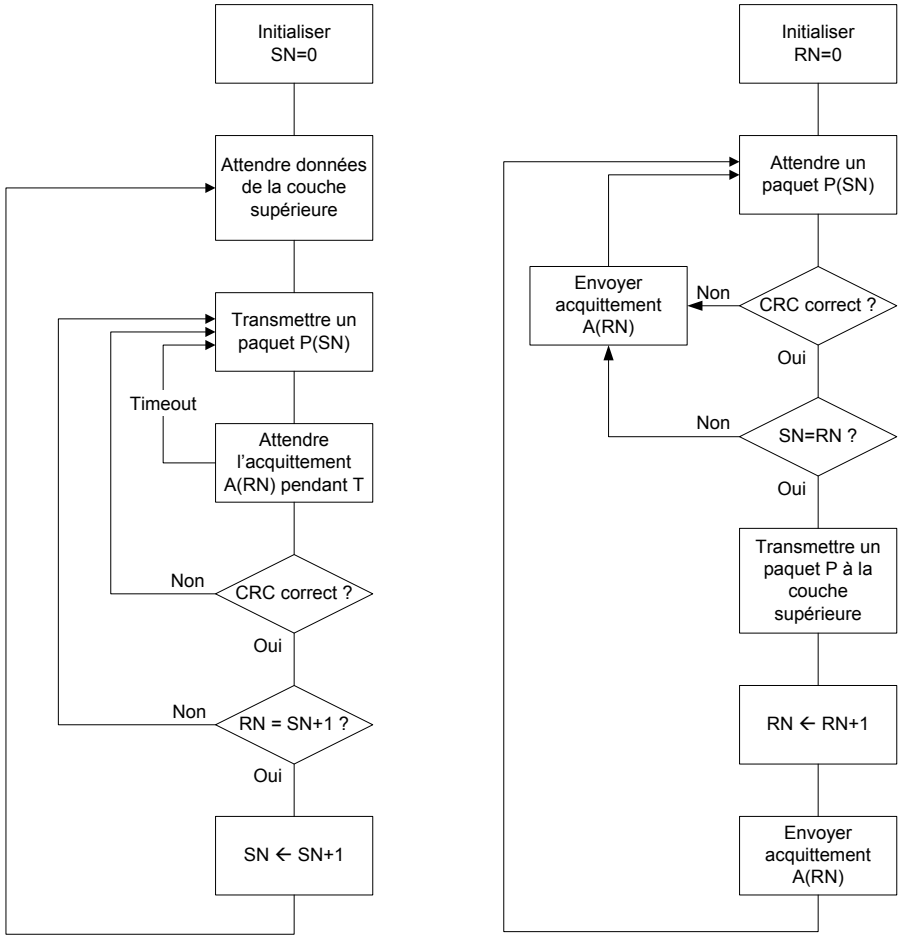


Figure 10: Algorithme « envoyer et attendre »

On peut voir que lorsque le récepteur reçoit un paquet qui n'est pas le paquet attendu, il le supprime et renvoie un acquittement avec le prochain numéro de séquence attendu. Cet acquittement a la fonction d'un acquittement négatif et contribue à la robustesse du protocole.

6.1.3 Taux d'utilisation du canal

La procédure « envoyer et attendre » exploite très mal la capacité offerte par le canal de transmission. L'envoi du prochain paquet ne peut commencer qu'après la réception de l'acquittement correspondant. Le temps entre l'émission du premier bit du paquet de données et la réception complète de l'acquittement est appelé temps de réponse ou

RTT (*Round Trip Time*). Il est donné par la somme des délais suivants (voir Figure 11) :

- Délai de transmission du paquet de données d_{data}. Il est calculé comme le quotient de la longueur du paquet en bits et le débit binaire du canal en bit/s : $d_{data} = l_{data} / C$.
- Délai de propagation du canal entre A et B : d_{prop}. Il est calculé comme le quotient de la distance entre A et B et la vitesse de propagation du signal.
- Délai de transmission de l'acquittement : $d_{ACK} = l_{ACK} / C$.
- Délai de propagation du canal entre B et A. Pour des raisons de simplicité nous supposons un canal symétrique qui montre le même délai de propagation dans les deux sens.

Nous omettons d'éventuels délai supplémentaires à cause du traitement des paquets sur les stations.

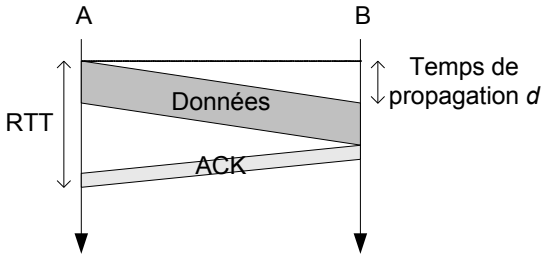


Figure 11: Temps de réponse dans « envoyer et attendre »

Le temps de réponse RTT est alors donné par :

$$RTT = (l_{data} + l_{ACK}) / C + 2d_{prop}.$$

Le **taux d'utilisation** ρ du canal, défini comme le rapport entre le débit effectif obtenu D_{eff} et la capacité C du réseau, peut être calculé comme suivant :

$$\rho = \frac{l_{data}}{RTT \cdot C} = \frac{l_{data}}{l_{data} + l_{ACK} + 2dC},$$

Le facteur $RTT \cdot C$ est appelé « **produit largeur de bande-délai** » (*bandwidth delay product*). L'utilisation du terme « largeur de bande » pour le débit binaire du canal est un abus de langage qui est très courant.

On s'aperçoit que l'utilisation du réseau, donc aussi le débit de transmission effectif, est inversement proportionnelle à ce produit largeur de bande-délai. Ceci est illustré à la Figure 12 pour un délai de propagation de 10ms, une taille des paquets et des acquittements de 1000 et 10 bits respectivement et une capacité entre 1000 bits/s et 1Mb/s.

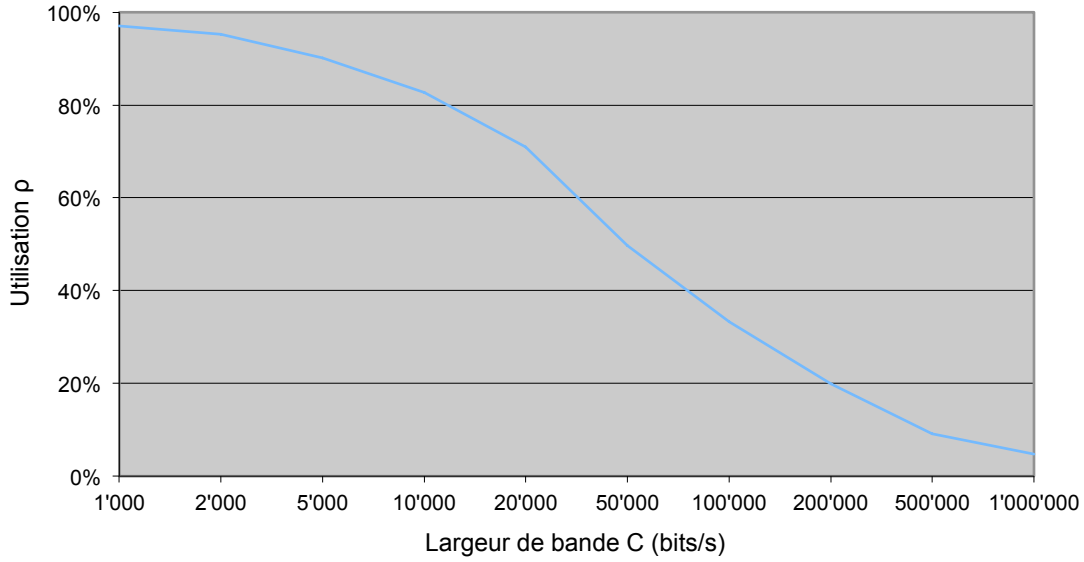


Figure 12: Efficacité du protocole « envoyer et attendre » en fonction du débit binaire C du canal

On voit que le taux d'utilisation du canal peut être très faible si le débit du canal est important et si le temps de propagation est élevé.

La procédure « envoyer et attendre » à la mérite d'être simple et robuste. Elle est encore utilisé dans des systèmes qui demandent une grande robustesse du protocole contre des erreurs de transmission (par exemple les réseau sans fil) et dans des réseau ou les stations n'ont pas la capacité d'exécuter des algorithmes plus complexes (par exemple pour le contrôle de machines dans les réseaux industriels).

La méthode « envoyer et attendre » permet d'ailleurs de réaliser un **contrôle de flux** de manière très simple. Lorsque l'émetteur et le récepteur ne travaillent pas à la même vitesse, un émetteur rapide peut surcharger un récepteur lent. Un contrôle de flux permet d'adapter la vitesse d'émission aux possibilités du récepteur. Dans la méthode « envoyer et attendre » il suffit pour le récepteur de retarder l'envoi de l'acquittement jusqu'au moment où il est prêt de traiter une nouvelle trame. Ainsi, il contrôle parfaitement la vitesse à laquelle il aimerait recevoir les données.

Dans les réseaux performants à haut débit, le protocole « envoyer et attendre » n'est pas assez efficace pour exploiter le canal. D'autres protocoles ont été développés qui permettent un meilleur taux d'utilisation : le protocole **Go-Back-n** et le **rejet sélectif**.

6.2 Go-back-n ARQ

Go-back-n est le protocole de retransmission automatique le plus répandu. Il est utilisé dans différents protocoles de liaison de données comme HDLC et LAPB, qui seront présentés plus tard dans ce chapitre. En outre, Go-back-n est aussi la base du contrôle de flux dans des protocoles de la couche transport, notamment de TCP.

L'idée principale de Go-back-n est très simple. Les paquets à transmettre sont numérotés séquentiellement par l'émetteur. Ce numéro de séquence SN est inclus dans l'entête du paquet avant d'envoyer le paquet sur la ligne. Contrairement à la

méthode « envoyer et attendre », l'émetteur n'attend pas l'acquittement d'un paquet avant d'envoyer le suivant. Il peut envoyer plusieurs paquets consécutifs, en anticipation des acquittements qu'il compte recevoir. L'émetteur doit évidemment conserver en mémoire une copie de tous les paquets envoyés mais pas encore acquittés, afin éventuellement de retransmettre ceux qui auraient subi une erreur de transmission.

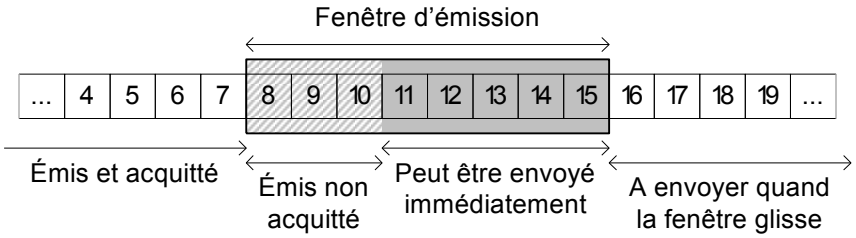
Le récepteur fonctionne essentiellement dans la même manière comme dans « envoyer et attendre ». Il accepte uniquement le paquet avec le prochain numéro de séquence attendu et renvoie un acquittement pour chaque paquet. L'acquittement contient le numéro de séquence *RN* du prochain paquet attendu. Lorsqu'un faux paquet arrive, il le supprime et renvoie un acquittement avec le numéro de séquence attendu.

Le paramètre *n* de ce protocole indique combien de paquets consécutifs peuvent être transmis sans attendre d'acquittement du récepteur. L'émetteur a l'intérêt de choisir un *n* petit, car il doit conserver une copie de tous les paquets en instance d'acquittement. Si ces derniers sont trop nombreux, la mémoire tampon aura une dimension exagérée. D'autre part, le risque d'avoir à procéder à des réexpéditions à la suite d'erreurs de transmission croît avec le nombre de paquets en instance d'expédition. On est donc amené à limiter le nombre de paquets qui peuvent être expédiés sans réception d'acquittements. Ceci est réalisé à l'aide d'une **fenêtre d'émission** dont la taille indique le nombre maximum de paquets qui peuvent être en instance d'acquittement. Les numéros de séquence SN des paquets dont l'expédition est autorisé sont définis par

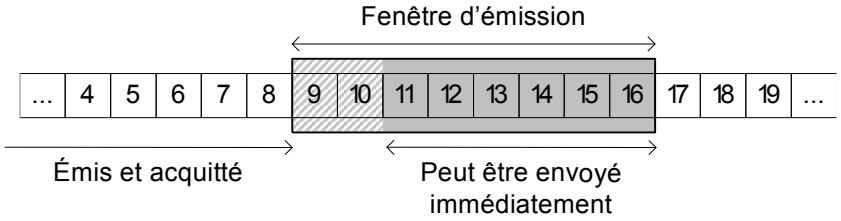
$$s \leq SN < s + n \,,$$

où *s-1* est le numéro de séquence du dernier paquet acquitté (donc le dernier acquittement reçu est *RN=s*). Le numéro de séquence *s* est la limite inférieure de la fenêtre. s+n est la limite supérieure, qui est le numéro du premier paquet dont l'expédition est interdite.

A un instant donné, la fenêtre peut être positionnée comme indiqué sur la Figure 13, avec une largeur de fenêtre *n*=8. Sur cet exemple, les paquets 8, 9 et 10 ont déjà été émis mais n'ont pas encore été acquittés. L'émetteur peut donc encore envoyer les paquets 11 à 15 avant de recevoir l'acquittement du paquet 8. Lorsque l'acquittement du paquet 8 est reçu, le tampon correspondant est libéré, et la fenêtre glisse d'une unité vers la droite, ce qui autorise l'expédition d'un paquet supplémentaire, ici du paquet 16. On voit donc que la fenêtre d'émission glisse vers la droite au fur et à mesure de la réception des acquittements, d'où le nom de **fenêtre coulissante** ou **fenêtre glissante** (*Sliding Window*) donné à la procédure Go-back-n.



a) Fenêtre d'émission



b) Après la réception d'un acquittement pour 8

Figure 13: Principe de la fenêtre glissante

6.2.1 Retransmissions

Il y a deux façons de déclencher la retransmission d'un paquet. Supposons d'abord une *gestion passive*, basée sur un temporisateur de retransmission. La Figure 14 montre la situation avec une taille de fenêtre de 4.

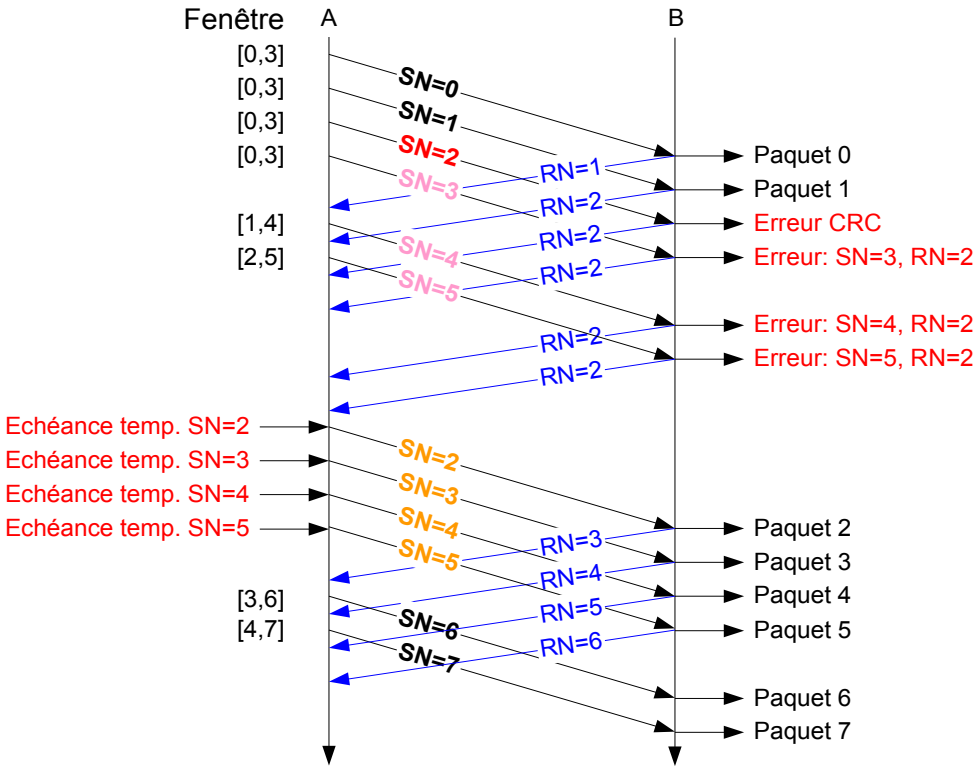


Figure 14: Retransmission basée sur les temporisateurs

Les paquets 0 et 1 sont reçus correctement et le récepteur les passe directement à la couche supérieure. Le paquet 2 par contre contient une erreur bit et le récepteur B le supprime. Il renvoie un acquittement qui contient le numéro de séquence RN=2 comme prochain paquet attendu. Les paquets 3, 4 et 5 sont reçus hors séquence et donc jetés. On voit donc que le destinataire n'a besoin ici que d'un seul tampon de réception.

Comme aucun acquittement valable n'est reçu pour le paquet 2, la retransmission est déclenchée par le temporisateur qui a été armé lors de l'émission de ce paquet. Les paquets suivants sont retransmission à cause de l'échéance de leurs temporisateurs respectifs.

On observe que le délai jusqu'au déclenchement des temporisateurs interrompe le flux de données de A à B. On peut donc améliorer l'utilisation de la ligne en adoptant une *gestion active*, où une station peut signaler immédiatement toute erreur qu'elle détecte, en expédiant un acquittement négatif (NAK). Comme dans la méthode « envoyer et attendre », au lieu de utiliser des paquets NAK, il est possible d'utiliser un seul type d'acquittement et ce le numéro de séquence contenu qui fournit toute l'information sur les erreurs de transmission. Cette méthode de déclencher une retransmission est illustrée sur la Figure 15.

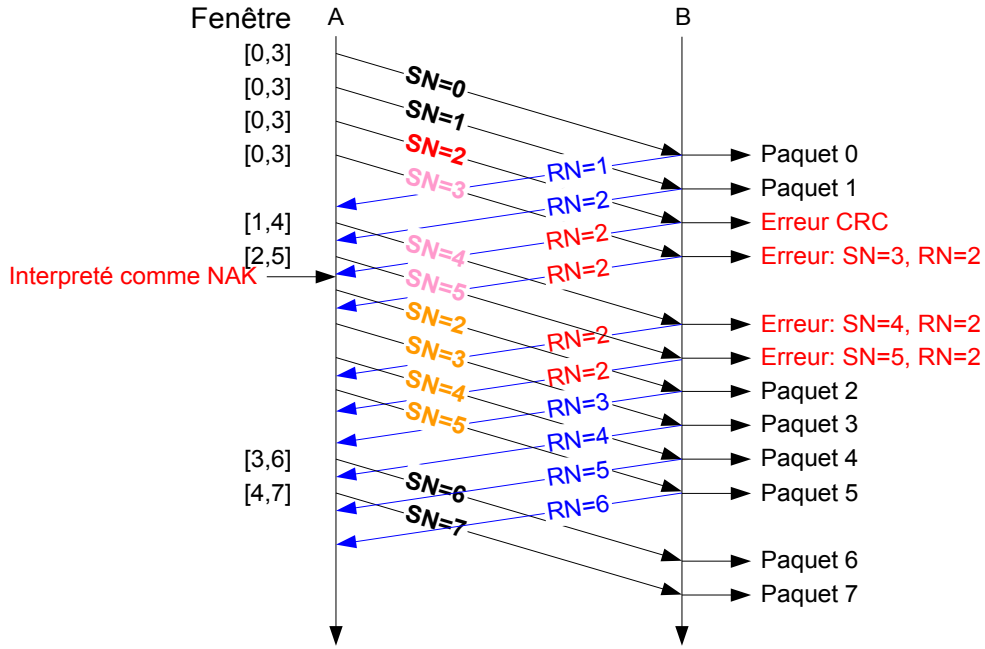


Figure 15: Retransmission déclenchée par un acquittement négatif

Quand l'émetteur reçoit un acquittement pour le paquet 2 au lieu du paquet 3 il en déduit qu'une erreur de transmission s'est produit. Il procède alors à la retransmission de tous les paquets à partir du paquet 2. On voit que le déclenchement par un acquittement négatif permet d'éviter l'attente jusqu'à l'échéance du temporisateur de retransmission et améliore ainsi l'utilisation de la ligne.

6.2.2 Numéros de séquence

Nous avons jusqu'ici supposé implicitement que la séquence des numéros des paquets est infinie, ce qui permet de repérer chaque paquet de façon unique et d'en assurer

l'acquittement sans ambiguïté. En pratique, il est évidemment impossible d'adopter une telle disposition, puisque ceci conduirait à prévoir dans les paquets et les acquittements des champs de dimension infinie pour les numéros de séquence. Afin de résoudre ce problème, on est donc amené à numéroter les paquets modulo un nombre *M*. La question qui se pose maintenant consiste à déterminer quelle est la valeur de *M*, compte tenu de la taille *n* de la fenêtre d'émission.

A chaque instant, un nombre de *n* paquets peuvent être en attente de l'acquittement. Afin qu'il n'y ait pas d'ambiguïté possible sur la nature de l'acquittement, il faut évidemment que les n+1 nombres allant de s à s+n soient tous distincts modulo M. Ceci impose la condition

$$n < M$$

pour une taille de fenêtre n. La Figure 16 montre la situation pour une taille de fenêtre de 4 et les numéros de séquence modulo 5. On peut voir qu'un acquittement comportant un numéro de séquence entre 1 et 4 est interprété sans ambiguïté. Un acquittement avec RN=0 accuse forcément la réception correcte du paquet 4, comme le paquet 0 (à gauche) est en dehors de la fenêtre, ce qui signifie qu'il a été acquitté correctement (par un acquittement RN=1) et ne sera plus redemandé par le récepteur.

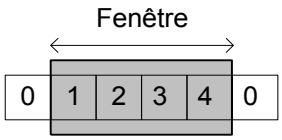


Figure 16: Numéros de séquence modulo 5 pour une taille de fenêtre de 4

On en déduit alors que la condition suffisante pour assurer l'interprétation correcte des acquittements est

$$M = n + 1$$

pour Go-back-n avec une taille de fenêtre de *n*.

6.3 Rejet sélectif

Avec les méthodes précédentes, le rejet d'un paquet provoque la retransmission de tous les paquets à partir de celui qui était en erreur. Cette politique est liée au fait que tous les paquets reçus hors séquence sont jetés par le récepteur, et elle conduit évidemment à un gaspillage de la capacité du canal, puisqu'il suffit d'une erreur que affecte uniquement le premier paquet d'un groupe pour provoquer la retransmission de tous la paquets d'un groupe. Plus précisément, les données retransmises correspondent au minimum

- De la taille de toute la fenêtre d'émission et
- du produit largeur de bande-délai (où le délai est le temps aller-retour) du canal.

Dans beaucoup de canaux de transmission, la probabilité d'une ou plusieurs erreurs dans un paquet et de 10^{-4} ou moins et dans un tel cas l'effet de la retransmission de plusieurs paquets au lieu d'un seul est minime. Pour des canaux avec un produit

largeur de bande – débit élevé et un taux d'erreur élevé (comme des liens satellite), la dégradation de l'utilisation du canal peut être sensible.

Dans ces cas, il est possible d'améliorer le taux d'utilisation de la ligne en remplaçant le rejet simple par un **rejet sélectif** (*Selective Repeat*). L'idée principale de cette méthode est que le récepteur accepte les paquets hors séquence et ne demande la retransmission que pour les paquets erronés ou perdus. Un nouveau type d'acquittement négatif, SREJ, est alors introduit, qui indique le numéro de séquence du paquet rejeté.

Dans ces conditions, les échanges sont organisés comme indiqué sur la Figure 17, où la réception d'un paquet incorrecte avec SN=1 provoque l'envoie d'un rejet sélectif SREJ=1 e la réexpédition du paquet erroné. Les paquets SN=2 et SN=3 reçus correctement par le destinataire avant le paquet retransmis sont stockés temporairement jusqu'à bonne réception de ce dernier, de façon à pouvoir être ensuite passés en séquence à la couche supérieure.

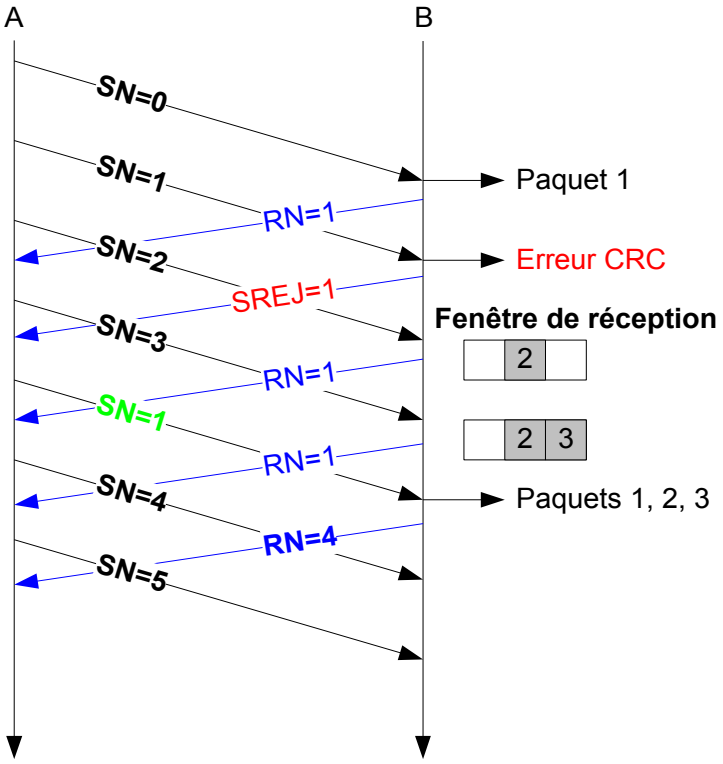


Figure 17: Principe du rejet sélectif

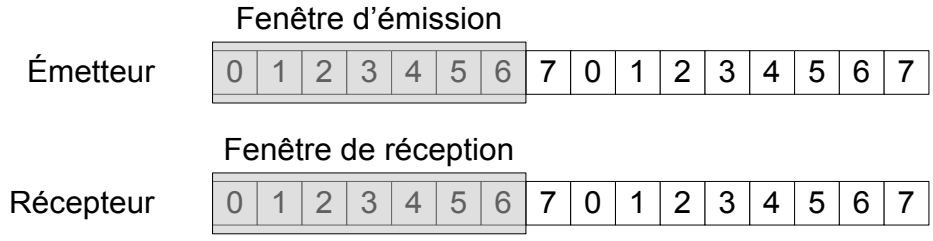
On voit donc qu'il faut prévoir un certain nombre de tampons de réception et un mécanisme de remise en séquence pour assurer le bon fonctionnement de la procédure. D'une manière symétrique à l'organisation adoptée au niveau de l'expéditeur, on définit une fenêtre de réception, qui définit la plage des numéros de séquence acceptables par le destinataire, et dont la valeur numérique indique le nombre de tampons de réception disponibles.

6.3.1 Numéros de séquence

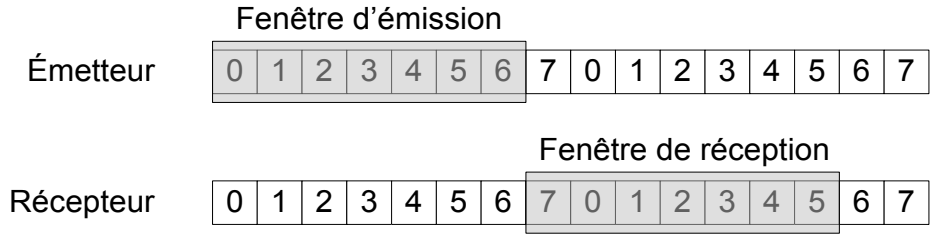
Recevoir les paquets d'une manière non séquentielle introduit de nouveaux problèmes. Nous allons illustrer ce propos par un exemple. Supposons que nous ayons

un numéro de séquence codé sur 3 bits (M=8) et que le récepteur utilise une taille de fenêtre d'émission de 7, conforme au critère trouvé pour Go-back-n.

Au départ, émetteur et récepteur sont dans l'état indiqué à la Figure 18a).



a) État initial



b) Après l'émission de 7 paquets et la perte des 7 acquittements

Figure 18: Rejet sélectif avec M=8 et une taille de fenêtre de 7

Les fenêtres permettent de transmettre les paquets 0 à 6 sans attendre d'acquittements. Si tous ces paquets sont correctement reçus, le récepteur les acquitte et sa fenêtre est mise à jour. Il peut alors accepter les paquets suivants 7, 0, 1, 2, 3, 4 et 5, comme indiqué à la Figure 18b). Supposons qu'il y ait des problèmes de transmission et que tous les acquittements soient perdus. Le temporisateur de l'émetteur expire et l'émetteur renvoie le paquet 0. Quand le paquet parvient au récepteur, celui-ci vérifie si son numéro de séquence se trouve dans la nouvelle fenêtre ; malheureusement, tel est le cas à la Figure 18b) et le paquet est accepté. Le récepteur envoie un acquittement avec RN=6, comme les paquets 0 à 6 ont été correctement reçus. L'émetteur, apprenant que tous les paquets précédemment transmis ont été bien reçus, met à jour sa fenêtre et envoie les paquets 7, 0, 1, 2, 3, 4 et 5. Le paquet 7 est accepté par le récepteur qui peut maintenant passer les paquets 7 et 0 à la couche supérieure. Finalement, la couche supérieure reçoit un mauvais paquet et le protocole se trouve pris par défaut.

Ce problème provient du fait que les numéros de séquence des deux fenêtres se superposent. Pour une taille de fenêtre *n*, l'émetteur a le droit d'émettre tous les paquets avec un numéro de séquence SN qui remplit

$$SN \in [SN_{\min}, SN_{\min} + n - 1],$$

où SN_{min} est le premier numéro de séquence dans la fenêtre d'émission. Après la réception du paquets SN_{min}+n-1, le récepteur peut accepter les paquets avec SN

$$SN \in [SN_{\min} + n, SN_{\min} + n + n - 1].$$

En fonction des acquittements reçus, le prochain paquet transmis par l'émetteur peut alors se trouver dans l'intervalle

$$SN \in [SN_{\min}, SN_{\min} + n - 1] \cup [SN_{\min} + n, SN_{\min} + n + n - 1] \leftrightarrow$$
$$SN \in [SN_{\min}, SN_{\min} + 2n - 1]$$

Le récepteur doit alors être capable de distinguer $2n$ numéros de séquence différents, donc si les numéros de séquence sont calculés modulo M, M doit remplir la condition

$$M \geq 2n$$

pour la méthode de rejet sélectif.

6.4 Efficacité des protocoles de retransmission

6.4.1 Taux d'utilisation de « envoyer et attendre »

La motivation d'utiliser des stratégies de retransmission plus complexes est d'améliorer le faible taux d'utilisation par rapport à « envoyer et attendre ». Comme montré ci-dessus, le taux d'utilisation ρ de « envoyer et attendre » est limité à

$$\rho = \frac{l_{data}}{RTT \cdot C} = \frac{l_{data}}{l_{data} + l_{ACK} + 2dC}.$$

Le terme RTT·C est appelé le « produit largeur de bande – délai » du canal.

6.4.2 Taille de fenêtre optimale pour Go-back-n et Rejet Sélectif

Nous voulons maintenant déterminer le taux d'utilisation d'un protocole à fenêtre glissante, donc de Go-back-n simple et avec rejet sélectif.

Considérons d'abord le cas sans erreur de transmission. La Figure 19 montre qu'une taille de fenêtre suffisante permet d'exploiter le canal de transmission (full-duplex) à 100%.

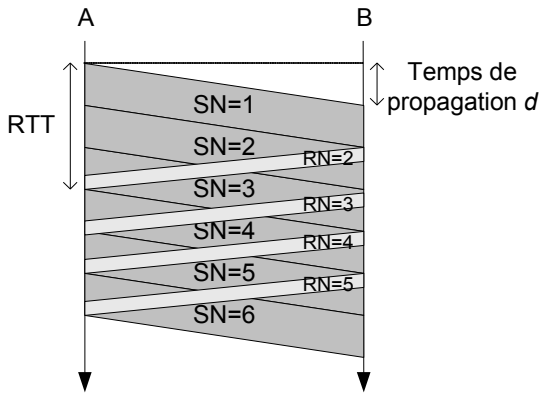


Figure 19: Une taille suffisante de la fenêtre glissante permet d'exploiter le canal de transmission à 100%

La taille n optimale de la fenêtre peut facilement être trouvée :

$$\rho \overset{!}{=} 1 = \frac{n \cdot l_{data}}{RTT \cdot C} \Rightarrow$$
$$n = \left\lceil \frac{RTT \cdot C}{l_{data}} \right\rceil .$$

Ainsi on obtient la règle simple :

Dans un protocole à fenêtre glissante, la taille de fenêtre optimale, mesuré en octets, doit être égale au produit largeur de bande – délai.

Avec cette taille de fenêtre, un taux d'utilisation de $\rho=1$ peut être atteint, mais seulement dans le cas idéalisé d'absence d'erreurs de transmission.

6.4.3 Taux d'utilisation de Rejet Sélectif avec erreurs

Après avoir obtenu ce résultat simple mais important, regardons le cas avec erreurs de transmissions. Considérons le cas d'un protocole de retransmission idéal, dans lequel seulement les paquets erronés sont retransmis. C'est le cas pour le protocole rejet sélectif si uniquement les erreurs dans les paquets de données sont considérées et les erreurs dans les acquittements sont négligées. Le nombre de paquets reçus est alors égal au nombre de paquets émis moins les paquets en erreur. Comme la probabilité d'un paquet d'être erroné est de p, on obtient :

$$n_{reçu} = n_{émis} - p \cdot n_{émis},$$

où $n_{reçu}$ est $n_{émis}$ sont les nombres de paquets reçus et émis, respectivement

Si l'on suppose que les paquets sont émis avec la vitesse maximum, alors le rapport entre le nombre de paquets reçus et le nombre de paquets émis (par unité de temps) donne l'utilisation du canal. Alors nous obtenons :

$$\rho \leq 1 - p.$$

Cette inégalité donne la limite du taux d'utilisation pour un protocole de retransmission idéal, qui ne retransmet que les paquets erronés. Uniquement les erreurs dans les paquets de données sont considérées, les erreurs dans les acquittements sont négligées.

6.4.4 Taux d'utilisation de « Go-back-n » avec erreurs

Dans le protocole « Go-back-n » avec une taille de fenêtre suffisante, les données à retransmettre après un seul paquet erroné correspondent au produit largeur de bande – délai $RTT \cdot C$ du canal. Pour la suite, nous définissons la constante β comme le nombre moyen de paquets que l'émetteur peut transmettre pendant un temps aller-retour : $\beta = \mathbf{min}\left(\left\lceil RTT \cdot C \middle/ l_{data} \right\rceil, n\right)$. Pour une fenêtre d'émission correctement dimensionnée, β est égal à n.

Dans l'analyse suivante, nous ne considérons que les erreurs dans les paquets de données et négligeons les erreurs dans les acquittements, qui sont en général beaucoup plus courts que les paquets de données et donc moins susceptibles aux erreurs.

Si un paquet est reçu sans erreur à la première transmission, un seul paquet et transmis dans ce cas. Cet évènement a une probabilité de (1-p). Si le paquet est erroné à la première transmission et reçu avec succès lors de la première retransmission, le nombre total de paquets transmis pour la réception avec succès est de $1+\beta$. Cet évènement a une probabilité de (1-p)·p. De manière analogue, si un paquet est reçu sans erreur après k retransmissions, le nombre total de paquets transmis et de $1+k\beta$ et la probabilité de cet évènement est de (1-p)·p^k.

L'espérance du nombre de paquets transmis par paquet reçu avec succès est donc :

$$E\left[\frac{n_{émis}}{n_{reçu}}\right] = \sum_{k=0}^{\infty}(1+k\beta)(1-p)p^k .$$

Après quelques transformations nous trouvons

$E\left[\frac{n_{émis}}{n_{reçu}}\right] = \frac{1-p+p\beta}{1-p}$. Si on suppose à nouveau que les paquets sont émis avec la vitesse maximum, l'inverse de ce rapport dans la vitesse maximum de réception de paquets, donc le taux d'utilisation effectif du canal :

$$\rho \le \frac{1-p}{1-p+p\beta}.$$

6.4.5 Exemple

A l'aide des résultats obtenus ci haut, nous allons maintenant évaluer les performances d'une liaison. Les caractéristiques de la liaison sont :

Débit binaire	C=1 Mb/s
Temps aller-retour	RTT=10 ms
Taux d'erreurs bit	BER=10^{-6}
Longueur de trame	l_{data}=1000 bits

Le taux d'erreurs bit nous permet de calculer la probabilité d'une trame erronée. La probabilité que la trame ne contienne pas d'erreurs est
$1-p=(1-BER)^{l_{data}} \approx 1-l_{data}\cdot BER$.
La probabilité d'une trame erronée peut donc être approchée par $p = l_{data}\cdot BER$.

La taille optimale de la fenêtre n peut être calculé comme :
$n = RTT \cdot C/l_{data} = 1\,\text{Mb/s}\cdot 10\,\text{ms}/1000\,\text{bits} = 10$. Le nombre β de paquets transmis par RTT vaut également 10.

« Envoyer et attendre »

En utilisant le protocole « envoyer et attendre » nous obtenons un taux d'utilisation

$\rho = \frac{l_{data}}{RTT\cdot C} = \frac{1000\,\text{bits}}{1\,\text{Mb/s}\cdot 10\,\text{ms}} = 0{,}1 = 10\%$ sans erreurs. Le taux d'utilisation effectif avec erreurs dépend de la valeur du temporisateur. Dans le cas idéal ou la perte d'un paquet serait immédiatement détecté on obtient un taux d'utilisation effectif de

$$\rho_{eff} = (1-p)\rho = \left(1-10^{-3}\right)\cdot 0{,}1 = 0{,}0999 = 9{,}99\%,$$

où p est à nouveau la probabilité d'une erreur lors d'une transmission.

« Rejet sélectif »

Pour un protocole de retransmission idéal, donc une version idéalisée du protocole « Rejet sélectif » le taux d'utilisation est limité à

$$\rho \le 1-p = 1-10^{-3} = 0{,}999 = 99{,}9\%.$$

On voit donc que l'effet de pertes de paquets est minime.

« Go-back-n »

Pour le protocole « Go-back-n » enfin, le taux d'utilisation est limité à

$$\rho \le \frac{1-p}{1-p+p\beta} = \frac{1-10^{-3}}{1-10^{-3}+10^{-3}\cdot 10} = \frac{0{,}999}{1{,}009} = 0{,}99 = 99\%.$$

Par rapport au protocole de retransmission idéale, 0,9% du débit du canal sont alors utilisés pour les retransmissions non nécessaires.

6.5 Gestion des acquittements

6.5.1 Groupement d'acquittements

Non seulement les paquets de données mais aussi les acquittements peuvent contenir des erreurs, quoique avec une probabilité plus faible comme les acquittement sont en générale beaucoup plus courts que les paquets de données. Une technique qui permet de minimiser l'effet d'acquittements perdus et ainsi d'améliorer la stabilité du protocole consiste à acquitter plusieurs paquets de données avec un seul accusé de réception. Dans cette méthode de **groupement des acquittements** ou **acquittements cumulatifs** il suffit de donner une nouvelle signification aux numéros de séquence de réception RN, en convenant qu'il acquitte tous les paquets de données dont le numéro de séquence d'émission SN est inférieur à RN. Cette technique réduit donc aussi le nombre d'acquittements expédiés sur la ligne.

6.5.2 Piggybacking

Dans beaucoup d'applications le flux de données est bidirectionnel. Le taux d'utilisation de la ligne peut encore être amélioré de façon supplémentaire en employant une méthode appelée ***piggybacking*** ou **superposition des acquittements**. En effet, rien ne s'oppose a priori à ce que des paquets de données soient acquittés par d'autres paquets de données transmis dans le sens opposé. Ceci peut être réalisé en insérant dans chaque paquet de données le numéro RN du prochain paquet de données attendu dans l'autre sens.

La technique d'encapsulation des acquittements n'est évidemment applicable que lorsque la station concernée dispose au moins d'un paquet de données en instance de transmission au moment où elle est invitée à expédier un acquittement. D'autre part, chaque paquet de données doit maintenant contenir un numéro de séquence réception

RN en plus du numéro de séquence émission SN, ce qui consomme quelques bits supplémentaires.

7 Contrôle de flux

Lorsqu'un émetteur émet de façon systématique plus de trames que le récepteur ne peut en accepter, le récepteur se verra dans une situation où il doit jeter des trames reçues comme il lui manque des ressources pour les traiter et stocker. C'est le cas lorsque l'émetteur est sur un ordinateur rapide (ou peu chargé) et que le récepteur est sur une machine lente (ou très chargée). La couche liaison doit alors mettre en œuvre un mécanisme pour éviter cette situation.

La solution consiste à instaurer un **contrôle de flux** pour contraindre l'émetteur à ne pas envoyer plus de trames que le récepteur ne peut en accepter. Le contrôle de flux est alors un protocole qui permet au récepteur final de signaler à l'émetteur s'il peut recevoir de nouvelles données ou non.

Une méthode souvent utilisée à la couche physique est le contrôle de flux logiciel, basé sur les caractères XON et XOFF. Lorsque le récepteur n'est temporairement pas en mesure d'accepter de nouvelles données, il en informe l'émetteur en envoyant un caractère XOFF. La réception de ce caractère par l'émetteur à l'effet similaire à la négation du signal CTS : l'émetteur cesse temporairement toute transmission. Une fois que le récepteur a résolu le problème, il envoie un caractère XON pour relancer la transmission des données.

Les stratégies de retransmission « envoyer et attendre », « Go-back-n » et « rejet sélectif » permettent non seulement de gérer les retransmissions mais aussi de contrôler, dans une certaine mesure, la vitesse de la transmission.

Le mécanisme le plus simple assure une **régulation par tout ou rien**, qui permet au destinataire de commander l'arrêt ou au contraire la reprise de l'envoie de trames par l'expéditeur. Dans la stratégie « envoyer et attendre » il suffit en principe que le récepteur retarde un acquittement d'une trame reçue pour empêcher l'émetteur de transmettre. Dès que le récepteur est en mesure de recevoir une nouvelle trame, il envoie l'acquittement et l'émetteur répondra avec la trame suivante.

Deux problèmes font que le retardement d'acquittement n'est utilise que rarement comme mécanisme de contrôle de flux. Premièrement, elle ne s'applique évidemment plus si l'émetteur utilise un temporisateur pour gérer les retransmissions des trames. Dans ce cas, l'acquittement doit arriver avant l'échéance du temporisateur pour éviter une retransmission non nécessaire. Cette contrainte ne laisse que peu de liberté au récepteur pour contrôler la transmission.

Deuxièmement, la régulation par tout ou rien est très mal adaptée à la transmission sur une ligne à longue distance où l'on utilise typiquement un protocole à fenêtre glissante pour améliorer l'utilisation de la ligne. En effet, l'expéditeur peut alors très bien encore envoyer un nombre important de trames avant de cesser la transmission en attendre d'un nouvelle acquittement. En plus, le problème du déclenchement d'une retransmission à cause d'un acquittement retardé est également présent dans un protocole à fenêtre glissante.

Deux solutions ont été adoptées par les différents protocoles pour améliorer ces défauts. La première solution consiste à permettre au récepteur de signaler son

incapacité de recevoir d'autres trames par un autre moyen que le retardement d'acquittements. Beaucoup de protocoles de la couche liaison ont introduit des trames de contrôle spéciales appelées RNR (pour *Receive Not Ready*) et RR (pour *Receive Ready*) dans ce but. La trame RNR dit à l'émetteur de cesser temporairement la transmission de nouvelles trames. La reprise de la transmission est provoquée par la réception d'une trame RR. Cette méthode est applicable aux protocoles « envoyer et attendre », « Go-back-n » et « rejet sélectif ».

La deuxième méthode adoptée n'est applicable qu'aux protocoles à fenêtre glissante (c'est-à-dire à « Go-back-n » et « rejet sélectif ») et consiste à varier la taille de la fenêtre. Nous avons vu ci haut (Section 6.4.2) que le débit effectif d'une transmission à fenêtre glissante dépend linéairement de la taille de la fenêtre, jusqu'à la taille optimale. En réduisant la taille de la fenêtre d'émission, le débit de transmission peut être adapté graduellement. Une taille de fenêtre 0 signifie l'arrêt temporaire de toute transmission, une taille égale à la taille optimale (c'est-à-dire au produit largeur de bande – débit) permet un débit maximum. Ce protocole peut être mis en œuvre en utilisant deux fenêtres : une **fenêtre d'émission**, gérée par l'émetteur, et une **fenêtre de réception**, gérée par le récepteur. Cette méthode est illustrée à la Figure 20. Le récepteur communique à l'émetteur à l'aide des acquittements l'espace restant dans sa fenêtre de réception. L'émetteur considère comme taille effective le minimum entre la fenêtre d'émission et la fenêtre de réception. Une fenêtre de réception de 0 arrête temporairement la transmission. Pour reprendre, le récepteur retransmet le dernier acquittement, mais avec un taille de fenêtre de réception non nulle. Le protocole le plus connu qui utilise cette méthode de contrôle de flux est le protocole TCP qui travaille à la couche transport.

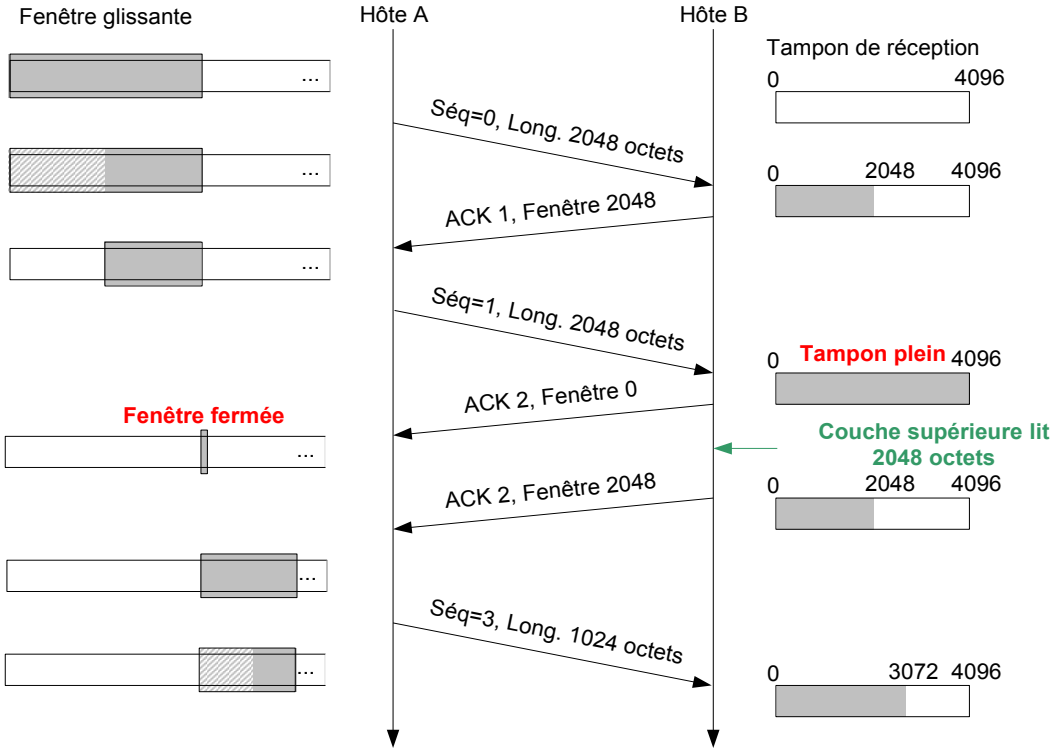


Figure 20: Contrôle de flux à l'aide d'une fenêtre glissante de taille variable

8 Bibliographie

Henri Nussbaumer, « Téléinformatique I » ; Presses polytechniques romandes ; Lausanne 1987. Chapitre 2.

Fred Halsall, “Data communications, Computer Networks and OSI”; 2nd edition; Addison Wesley; 1988. Chapitres 2 et 3.

Dimitri Bertsekas et Robert Gallager, “Data Networks”; 2nd edition; Prentice Hall; 1992. Chapitre 2.2.