Problèmes et exercices

Exercice 1 La notion de décibel

Énoncé

Dans un environnement urbain, la puissance sonore produite par les nombreuses sources de bruits est évaluée en décibels, en comparant la puissance sonore de la source de bruit à un niveau sonore de référence.

a Si on évalue la puissance sonore *S* d'une grosse moto à 87 dB, quelle est, en décibels, la puissance sonore produite par une bande de 8 motards roulant sur des motos identiques circulant à la même vitesse ?

b Trouvez la puissance sonore réellement émise.

Solution

a La bande de motards produit 8 fois plus de puissance sonore qu'une seule moto. On a donc : $10*\log_{10}(8S) = 10*\log_{10} 8 + 10*\log_{10} S$, ce qui revient à *ajouter* 10 fois le logarithme décimal de 8 au bruit d'une moto pour obtenir le nombre de décibels produit par les 8 motos.

Puisque : $10*\log_{10}8 = 10*\log_{10}2^3 = 3*10*\log_{10}2 = 9$ dB, la puissance des 8 motos vaut : $S = 87 + 9 = 96$ dB.

b Cela correspond à une puissance sonore de $4*10^9$, soit 4 milliards de fois le fond sonore de référence !

Remarque

Pendant que la valeur en décibels du bruit a augmenté d'environ 10 %, la puissance sonore réellement émise a été multipliée par 8.

Exercice 2 Évaluation d'un rapport signal/bruit (S/B)

Énoncé

Sur un support de transmission, le rapport *S/B* vaut 400.

a Quelle est la valeur de ce rapport en décibels ?

b Même question avec un rapport *S/B* de 40 000.

c Quelle est la valeur *N* en décibels d'un rapport *S/B* égal à 500 000 ?

Solution

a Un rapport S/B de 400 correspond à $10*\log_{10}400 : 10*(\log_{10}4 + \log_{10}100)$.
D'où : $20*(\log_{10}2 + \log_{10}100) = 26$ dB.

b Le rapport S/B est 100 fois plus élevé que le précédent, c'est-à-dire qu'il vaut : $26 + 20 = 46$ dB.

c On peut calculer simplement une bonne valeur approchée du nombre *N* de décibels en remarquant que : $500\,000 = 10^6 \div 2$. On aura donc :
$N = 10*(\log_{10}10^6 - \log_{10}2) = 10*[6*\log_{10}10 - \log_{10}2] = 60 - 3 = 57$ dB.

Exercice 3 Débit binaire et rapidité de modulation

Énoncé

Soit un signal numérique dont la rapidité de modulation est 4 fois plus faible que le débit binaire.

a Quelle est la valence du signal ?

b Si la rapidité de modulation du signal vaut 2 400 bauds, quel est le débit binaire disponible ?

Solution

a D'après la formule $D = R\ log_2 V$, nous trouvons : $D/R = log_2 V$ soit : $V = 2^{D/R}$, c'est-à-dire que la valence vaut 16.

b En appliquant la même formule, nous trouvons : $D = 2\ 400*4 = 9\ 600$ bit/s.

Exercice 4 Signaux transmis en bande de base et par modulation

Énoncé

Soit la suite d'éléments binaires `0 1 1 1 1 1 1 0`.

a Représentez les signaux transmis lorsqu'on transmet en bande de base avec les codes NRZ et Manchester.

b Représentez les signaux transmis lorsqu'on transmet les données avec une modulation d'amplitude à deux valeurs, une modulation de phase à deux valeurs, une modulation de fréquence à deux valeurs.

c Si le débit D est connu, quelle est la rapidité de modulation R ?

Solution

a Les figures 1.12 et 1.13 représentent les données codées en NRZ et Manchester :

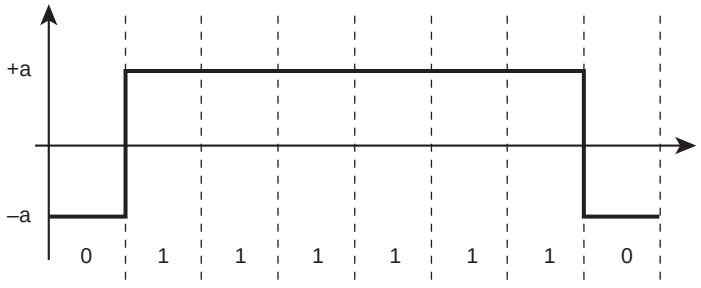


Figure 1.12
Codage NRZ.

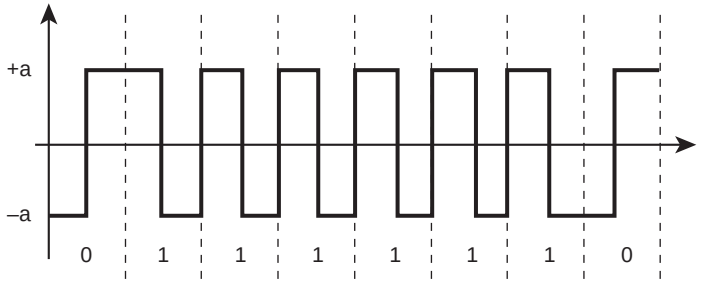
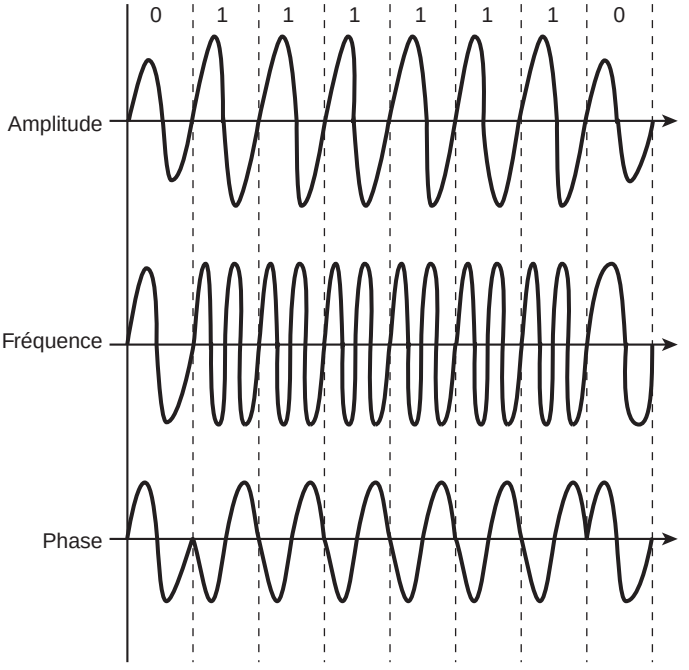


Figure 1.13
Codage biphase ou Manchester.

b Les modulations d'amplitude, de fréquence et de phase sont représentées à la figure 1.14.

Figure 1.14

Représentation des différentes modulations.



c Si D est connu et que la valence des signaux est égale à 2, alors $R = D$ bauds.

Exercice 5 Code Manchester et autres codes

Énoncé

Le code Manchester présente l'intérêt de posséder au moins une transition du signal au milieu de l'intervalle pour une bonne synchronisation du récepteur mais il peut présenter trop de transitions, en particulier si la suite de données binaires contient une longue suite de 0 par exemple.

a Représentez le signal transmis avec le code Manchester pour les données `100000000001`.

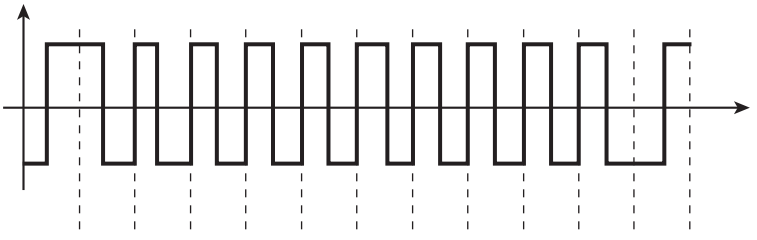
b Le code de Miller offre une alternative intéressante. Il consiste, à partir du code Manchester, à supprimer une transition sur deux. Dessinez le signal transmis pour les mêmes données et montrez que le décodage n'est pas ambigu.

Solution

a La figure 1.15 représente les données avec le code Manchester.

Figure 1.15

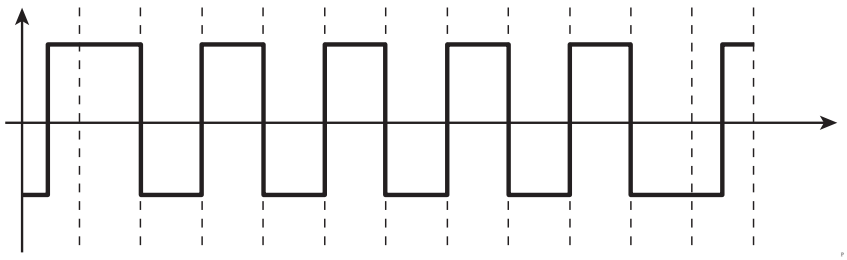
Données en codage Manchester.



b La figure 1.16 représente les données avec le code de Miller.

Figure 1.16

Données en codage de Miller.



Le décodage du code de Miller est très simple : une transition au milieu de l'intervalle représente un 1, une absence de transition dans l'intervalle représente un 0. Il n'y a donc aucune ambiguïté de décodage.

Remarque

Le choix d'un « bon » code est difficile ! Il faut trouver un compromis entre le nombre de transitions indispensable à la synchronisation du codec récepteur et une solution transparente aux données transmises.

EXERCICE 6 FORMULE DE SHANNON

Énoncé

Si on n'utilise pas de techniques de compression de données, une transmission de voix numérisée nécessite un débit binaire de 64 kbit/s.

a En supposant que la transmission se fasse par des signaux modulés de valence 32, quelle est la bande passante disponible, sachant que celle-ci est égale à la moitié de la rapidité de modulation utilisée ?

b Quel doit être le rapport S/B de la ligne de transmission offrant un débit binaire de 64 kbit/s et possédant une largeur de bande trouvée dans la question précédente ? On exprimera cette valeur en vraie grandeur et en décibels.

Solution

a On utilise la formule $D = R^{*}log_2V$.

On obtient : $64^{*}10^3 = R^{*}\log_2 32$, ce qui donne $D = 5R$, d'où : $R = 12\,800$ bauds. La bande passante est donc égale à 6 400 Hz.

b En utilisant la formule de Shannon $D = W^{*}log_2(1 + S/B)$, on trouve : $64^{*}10^3 = 6\,400^{*}\log_2(1 + S/B)$, d'où : $\log_2(1 + S/B) = 10$, c'est-à-dire que $S/B = 2^{10} - 1$, soit 1 023 (on pourra négliger le 1 devant le rapport S/B), ce qui correspond à 30 dB environ.

Exercice 7 Connexion à Internet

Énoncé

Pour vous connecter à Internet, vous avez relié votre ordinateur portable au réseau grâce à un modem de type PCMCIA, raccordé à la ligne téléphonique de votre domicile. On suppose que votre modem a un débit maximal de 56 kbit/s et que votre ligne téléphonique possède une bande passante comprise entre 300 et 3 400 Hz. Pendant votre connexion, vous constatez que la vitesse de transfert des données effective est 6 200 octet/s.

a Si la vitesse constatée ne provient que d'un mauvais rapport *S/B* de votre ligne, quelle est la valeur de ce rapport durant votre connexion ?

b La vitesse de transmission est maintenant de 24 800 bit/s. Si la rapidité de modulation est de 4 800 bauds, quelle est la valence du signal modulé ?

c On suppose que la ligne téléphonique répond au critère de Nyquist et que la rapidité de modulation vaut 4 800 bauds. Si on utilise la rapidité de modulation maximale, quelle est la bande passante du support ?

d Supposons que le débit binaire indiqué reste constant et égal à 49 600 bit/s pendant toute la durée de la connexion. Combien de temps devrez-vous rester connecté pour télécharger un fichier de 2 Mo (on pourra prendre ici 1 Mo = 10^6 octets) sur votre portable ?

e Vous utilisez désormais une connexion à 10 Mbit/s. Combien de temps resterez-vous connecté pour télécharger le même fichier que celui de la question d ?

Solution

a Le débit binaire de la ligne vaut 49 600 bit/s. D'après le théorème de Shannon, on obtient : 49 600 = 3100*$\log_2(1 + S/B)$, soit : $\log_2(1 + S/B) = 16$, d'où : $S/B = 2^{16} - 1$. En négligeant le 1, nous trouvons un rapport $S/B = 65536$, soit environ 48 dB.

b Toujours en utilisant le théorème de Shannon, nous trouvons : 24 800 = 3100*$\log_2(1 + S/B)$, soit : $S/B = 2^8 - 1 = 255$. Le rapport *S/B* vaut environ 24 dB.

c Selon le critère de Nyquist, la rapidité de modulation maximale est égale à 2 fois la bande passante de la ligne. Celle-ci vaut donc 2 400 Hz.

d Le temps *t* nécessaire pour transférer 2*10^6 octets est égal à : $t = 2*8*10^6/49\,600 = 322{,}58$ s soit environ 5 minutes et 22 secondes.

e Le temps *t* nécessaire n'est plus que de 1,6 s…

Exercice 8 Caractéristiques des modems V23 et V29

Énoncé

a Exprimez et comparez les valeurs du débit binaire et de la rapidité de modulation du modem *V23* et du modem *V29*. Le modem *V23* fonctionne à 1 200 bit/s avec une modulation de fréquences à deux valeurs. Le modem *V29* offre un débit binaire de 9 600 bit/s et utilise une modulation combinée d'amplitude et de phase (modulation d'amplitude à 2 valeurs et modulation de phase à 8 valeurs).

b Proposez un codage simple des données binaires transmises par le modem *V29*.

Solution

a Le modem normalisé *V23* est le « vieux » modem intégré au Minitel. Les caractéristiques techniques fournies montrent qu'il transmet des signaux de valence 2, c'est-à-dire qu'un intervalle de temps de 1/1 200 s contient 1 bit. Donc la rapidité de modulation de ce modem est égale à son débit binaire soit 1 200 bauds.

Dans le modem V29, on utilise deux amplitudes *A1* et *A2* et huit phases *P1*, *P2*, *P3*, *P4*, *P5*, *P6*, *P7*, *P8*. Pendant un intervalle de temps, il s'agit de la combinaison d'une amplitude et d'une phase, donc le modem transmet une valeur parmi les 16 possibles. Il transmet 4 bits par intervalle de temps ; les informations à transmettre sont codées par groupes par 4 bits (appelés parfois quadribits) par le modem. Voici un exemple possible de codage des quadribits :

`0000` ==> *A1* et *P1*	`1000` ==> *A2* et *P1*
`0001` ==> *A1 et P2*	`1001` ==> *A2* et *P2*
`0010` ==> *A1* et *P3*	`1010` ==> *A2* et *P3*
`0011` ==> *A1* et *P4*	`1011` ==> *A2* et *P4*
`0100` ==> *A1* et *P5*	`1100` ==> *A2* et *P5*
`0101` ==> *A1* et *P6*	`1101` ==> *A2* et *P6*
`0110` ==> *A1* et *P7*	`1110` ==> *A2* et *P7*
`0111` ==> *A1* et *P8*	`1111` ==> *A2* et *P8*

b Comme le débit du modem *V29* est de 9 600 bit/s, l'intervalle de temps est de 4/9 600 s, soit 1/2 400 s ; la rapidité de modulation vaut : 9 600/4 = 2 400 bauds. On peut retrouver ce résultat en appliquant la formule : $D = R^{*}log_2V$, dans laquelle D et V sont connus et valent respectivement 9600 et 16.

Exercice 9 Modem normalisé V32

Énoncé

Vous avez déniché dans votre cave un vieux modem fonctionnant à l'alternat, capable d'envoyer et de recevoir les données à 9 600 bit/s. Sans connaître les normes utilisées dans la construction de ce modem, vous essayez de trouver ce que pourraient être sa rapidité de modulation et la valence des signaux qu'il produit, sachant que la bande passante du téléphone vaut 3 100 Hz. Indiquez les solutions que vous avez trouvées.

Solution

La seule chose certaine est que la valence du signal produit doit être supérieure à 2 puisque, d'après le critère de Nyquist, le modem ne pourrait envoyer (ou recevoir) que 6 200 bit/s au maximum avec cette valence.

En appliquant la formule liant la rapidité de modulation au débit binaire et à la valence, vous obtenez : $\log_2 V = 9600/3100$, soit $V = 2^{3,097}$ environ. Sans même faire le calcul, vous vous rendez compte que cette solution est inacceptable puisque, par définition, la valence est un nombre entier. D'autre part, le débit binaire ne vaudra pas *exactement* 9 600 bit/s ! Il faut donc que la rapidité de modulation soit un sous-multiple entier du débit binaire, c'est-à-dire que le rapport entre les deux grandeurs doit être une puissance de 2.

Vous proposez :

a. Une rapidité de modulation valant 2 400 bauds. D'où : $D = 4R$ et donc $V = 16$.

b. Une rapidité de modulation valant 1 200 bauds. D'où : $D = 8R$ et donc V = 256.

c. Une rapidité de modulation valant 3 200 bauds. D'où : $D = 3R$ et donc V = 8.

En fouillant dans les vieilles normes AFNOR (Association française de normalisation), vous constatez que les modems transmettant à l'alternat à 9 600 bit/s fonctionnaient conformément à la recommandation *V32*. Cette norme préconisait une rapidité de modulation de 2 400 bauds ; la modulation employée était une modulation d'amplitude complexe utilisant une valence 16 ou 32.

Remarque

La valence 32 fournit ici un débit binaire de 5*2 400 = 12 000 bit/s soit plus que 9 600 ! En fait, sur les 12 000 bits transmis à la seconde, seuls 9 600 étaient réellement « utiles », les 2 400 autres servant de protection contre les erreurs : ils étaient calculés à partir des 9 600 premiers et permettaient au modem récepteur de détecter et de corriger d'éventuelles erreurs. Pour l'utilisateur, le débit réellement utilisable reste de 9 600 bit/s. Ce mécanisme, très général, sera repris au chapitre 2.

Exercice 10 Système de radiomessagerie

Énoncé

Un système de radiomessagerie de poche (un *pager*) répondant à la norme ERMES (*European Radio Message System*) présente les caractéristiques techniques suivantes :

- bande de fréquences : 169,425 MHz – 169,800 MHz ;
- modulation de fréquences à 4 états ;
- rapidité de modulation : 3 125 bauds ;
- rapport *S/B* d'un récepteur : 76 dB.

a Quel est le débit binaire réellement utilisé dans cette radiomessagerie ?

b En supposant qu'on transmette un octet par caractère, combien de temps faut-il pour transmettre un message de 200 caractères sur un récepteur de radiomessagerie ?

c Au lieu du débit binaire trouvé à la question a, quel débit binaire pourrait-on théoriquement obtenir en exploitant au mieux les caractéristiques techniques de la radiomessagerie ?

d Pourquoi n'est-ce pas utilisé ?

Solution

a Le débit binaire réellement utilisé est : $D = 3\,125*2 = 6\,250$ bit/s.

b Il faut : 8*200/6250 = 0,256 seconde pour transférer le message sur le récepteur.

c La bande passante du support vaut : $(169{,}8 - 169{,}425)*10^6 = 375$ kHz. D'après le théorème de Shannon, on pourrait transmettre au maximum : $D = 375*10^3*\log_2(1 + S/B)$ soit environ : 9 467 495 bit/s.

d Parce que la vitesse d'affichage utilisée est bien suffisante pour un lecteur humain, puisqu'un écran entier s'affiche en un quart de seconde. On peut ainsi se contenter d'employer des composants bon marché pour la fabrication des récepteurs.

EXERCICE 11 CODAGE DES INFORMATIONS

Énoncé

On utilise un alphabet de 26 caractères différents. Pour transmettre ces données, on code chaque caractère par une suite de bits.

a Si le codage des caractères est à longueur constante, combien faut-il de bits pour coder un caractère de cet alphabet ?

b Dans le réseau Télex (réseau télégraphique), on utilisait un alphabet contenant 5 bits par caractère. Comment pouvait-on coder les lettres de l'alphabet latin, les chiffres et d'autres symboles (comme les signes de ponctuation, par exemple) ?

c Combien de caractères différents peut-on représenter avec la méthode de codage précédente ?

Solution

a Il faut coder chaque caractère de l'alphabet avec un nombre constant de bits. 26 étant un nombre compris entre 16 et 32, on choisira donc la puissance de 2 par excès qui permet de coder tous les caractères, même si certains codages sont inutilisés. Il faut donc $\log_2 32$ bits pour coder les caractères de l'alphabet, c'est-à-dire 5 bits.

b Avec 5 bits, on peut coder 2^5 symboles soit 32 caractères différents, ce qui est notoirement insuffisant pour coder les 26 lettres, plus les chiffres et les signes de ponctuation. Les télégraphistes ont donc inventé la notion de *caractère d'échappement*, un caractère dont la présence modifie l'interprétation du ou des caractères qui suivent. On a défini un code « Lettre » et un code « Chiffre », les autres caractères étant interprétés en fonction du caractère d'échappement qui les précède. Ainsi, chaque codage binaire a deux interprétations, selon qu'on se trouve en mode « Lettre » ou en mode « Chiffre » (par convention, on reste dans le mode sélectionné tant qu'on ne trouve pas un nouveau caractère d'échappement).

c On dispose ainsi de 30 caractères en configuration « Lettre » et 30 caractères en configuration « Chiffre ». On dispose donc au maximum de 62 codes différents pour représenter tous les caractères.

Remarque

Pour la saisie des caractères dactylographiés sur un clavier, la touche Maj est un caractère d'échappement qui modifie la valeur du caractère suivant : tant qu'on n'appuie pas sur cette touche, on tape les minuscules ou les caractères spéciaux correspondant au codage en mode Minuscule.

Exercice 12 Interface ETTD-ETCD

Énoncé

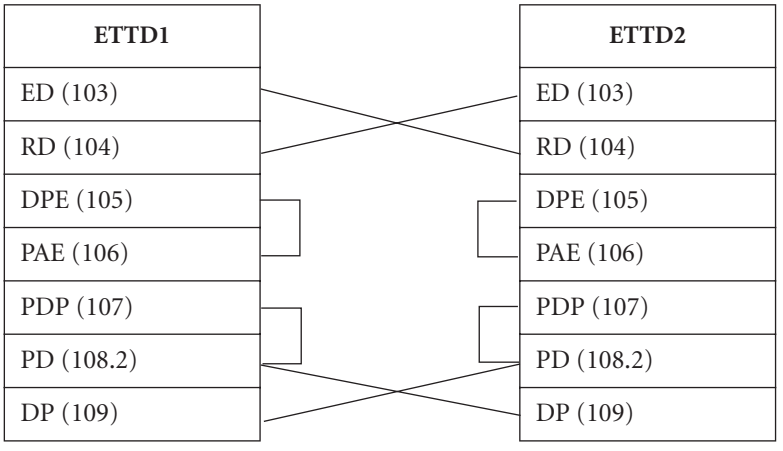
L'interface ETTD-ETCD définie par *V24* est conçue de telle sorte qu'un ETTD ne puisse être relié qu'à un ETCD. Peut-on relier deux ETTD utilisant localement (c'est-à-dire sans ETCD) une interface *V24* ? Reliez les signaux des circuits des deux ETTD donnés sur la figure pour permettre un échange de données correct entre les deux ETTD.

ETTD1
ED (103)
RD (104)
DPE (105)
PAE (106)
PDP (107)
PD (108.2)
DP (109)

ETTD2
ED (103)
RD (104)
DPE (105)
PAE (106)
PDP (107)
PD (108.2)
DP (109)

Solution

Oui, on appelle une telle liaison « zéro-modem » (ou *null modem*). Cela consiste à croiser les fils, de telle sorte que ce qui est émis par l'ETTD1 soit reçu par l'ETTD2 et *vice versa*. Les ETCD étant absents, le câblage doit toujours être prêt à émettre, d'où la boucle locale des circuits 105 – 106 de chaque côté, de même pour les circuits 107 et 108. Enfin, il n'y a plus de porteuse mais les deux ETTD doivent être informés du fonctionnement de l'interface. Les circuits 107 et 109 doivent recevoir un signal pendant la durée des échanges : on utilise pour cela le signal 108.



EXERCICE 13 PRINCIPES DE FONCTIONNEMENT DE L'ADSL

Énoncé

Examinons les principes de transmission utilisés dans l'ADSL. Dans la modulation DMT, la plage des fréquences disponible sur la boucle locale est divisée en 256 canaux juxtaposés de 4 312,5 Hz chacun. Le canal 0 est utilisé pour le téléphone vocal et les canaux 1 à 5 ne sont pas exploités pour éviter les interférences entre la voix et les données. Parmi les canaux restants, deux sont réservés pour le contrôle des flux montant et descendant, le reste est utilisé pour transmettre les données.

a Combien reste-t-il de canaux à utiliser pour le transfert des données dans les deux sens en modulation DMT ?

b De quoi dépend le nombre de canaux à affecter aux données de chaque sens ? Qui se charge de l'affectation des canaux ?

c Que faudrait-il faire pour que les flux montant et descendant aient des débits identiques ?

d L'utilisation la plus courante en ADSL consiste à réserver 32 canaux pour le flux montant et les canaux restants pour le flux descendant. Quel est le débit théorique que l'on peut obtenir pour le flux montant si l'on transmet des signaux binaires sur chaque canal ?

e Même question pour le flux descendant.

f Une autre technique de modulation utilise, pour le flux descendant, une rapidité de modulation de 4 000 bauds et émet 15 bits par signal transmis sur 224 canaux. Quel débit binaire peut-on obtenir avec cette technique ?

Solution

a Il reste 248 canaux pour les flux de données montant et descendant.

b Le nombre de canaux affectés à chaque sens dépend du débit binaire qu'on veut offrir aux abonnés : plus ce nombre est grand et plus le débit binaire sera important pour le flux considéré. C'est bien évidemment le fournisseur d'accès qui répartit les canaux, en allouant généralement 90 % des canaux au flux descendant et les 10 % restants au flux montant.

c Il faut simplement allouer autant de canaux pour le flux montant que pour le flux descendant. On obtient ainsi une technologie DSL symétrique (SDSL).

d On peut obtenir : 4 312,5*32 = 138 kbit/s pour le flux montant.

e Il reste pour le flux descendant : 248 – 32 = 216 canaux, soit un débit binaire de 931,5 kbit/s.

f On peut obtenir : 15*4 000*224 = 13,44 Mbit/s.

Remarque

Les technologies symétriques sont réservées aux opérateurs et aux fournisseurs d'accès. Elles ne sont pas disponibles pour les abonnés. On n'atteint pas dans la pratique le débit obtenu à la question f, car le rapport *S/B* des lignes est insuffisant le plus souvent. On obtient couramment 8 Mbit/s sur de courtes distances, avec une boucle locale de bonne qualité.

Chapitre 2

Les protocoles de liaison de données

Le circuit de données pouvant altérer les informations transportées, le *protocole de liaison de données* le supervise et définit un ensemble de règles pour assurer la fiabilité des échanges sur une *liaison de données*.
Ce protocole spécifie le format des unités de données échangées (les trames), leur délimitation, les moyens de contrôler leur validité (parité, code polynomial…), ainsi que le mode de correction des erreurs détectées. Il fixe également les règles du dialogue entre les deux extrémités de la liaison. Il exerce en outre deux fonctions importantes : le contrôle de flux (mécanisme vérifiant le rythme d'envoi des informations) et la gestion des acquittements (mécanisme validant la réception des informations).
HDLC (*High level Data Link Control*) est un exemple de protocole normalisé très répandu, orienté bit, transparent à tous les codes, dans lequel toutes les trames ont le même format. Il permet d'exploiter une liaison bidirectionnelle simultanée avec contrôle d'erreurs, de séquence et de flux.
PPP (*Point to Point Protocol*) en est une version très simplifiée, utilisée dans Internet.

1 Rôle et fonctions d'un protocole de liaison

Pour faire communiquer des machines identifiées par leurs adresses, il faut définir un grand nombre de règles concernant la structuration du dialogue, le format des messages transmis, leur enchaînement logique, le codage de l'information, le rythme de transmission, etc. L'ensemble des règles, assimilables à des règles d'orthographe et de grammaire définissant la construction des phrases d'une langue, s'appelle *protocole de liaison de données* ou *protocole de communication*. Un programme (*logiciel de communication*), installé sur les équipements qui doivent communiquer à distance, l'exécute. Afin d'assurer un maximum d'interopérabilité entre équipements différents, les instances de normalisation ont travaillé à la définition des protocoles de communication à l'échelle internationale.

Définition

Un *protocole* est un ensemble de règles et de formats de données à respecter pour échanger des données dans de bonnes conditions entre deux équipements ou deux programmes. Un protocole *de liaison de données* a pour objet de rendre fiable le circuit de données.

Alors que le circuit de données transmet des éléments binaires, le protocole de liaison de données travaille sur des blocs d'éléments binaires appelés *trames*. La trame est donc l'unité de données qu'il gère. Elle transporte les données de l'utilisateur et contient, en outre, des informations de commande, nécessaires au protocole pour garantir le bon déroulement du dialogue (certaines trames, les trames de *supervision*, sont d'ailleurs réduites aux seules informations de commande). Une trame compte différents *champs*. Chacun d'eux est un bloc d'éléments binaires dont la signification et l'interprétation sont précisées dans la définition du protocole.

Le protocole doit également définir les règles du dialogue et spécifier la façon de corriger les erreurs détectées. Enfin, on doit pouvoir détecter les pannes des équipements ou les ruptures complètes de liaison pour avertir l'utilisateur de l'indisponibilité du service.

Remarque

Définir un protocole de liaison de données consiste à préciser : le format des trames échangées, les conditions de délimitation des trames (début et fin) et leur validité, la position et la signification des différents champs d'une trame, la technique de détection d'erreur utilisée, les règles du dialogue (supervision de la liaison) et les procédures à respecter après détection d'erreurs ou de panne de la liaison.

1.1 Mise en forme des données

En théorie, les délimitations de début et de fin de trame sont indépendantes de la technique de transmission utilisée. En pratique, certains procédés utilisent des particularités du codage en ligne pour délimiter les trames. Les solutions les plus fréquentes sont la délimitation par une séquence binaire spéciale ou l'indication explicite de la longueur de la trame.

Délimitation par une séquence spécifique d'éléments binaires

Les trames ayant un nombre quelconque de bits, une séquence spécifique, appelée *fanion (*ou *flag)*, sert à indiquer le début aussi bien que la fin des trames. En général, il se compose de l'octet `01111110`. Un mécanisme de transparence est nécessaire pour éviter de retrouver cette séquence à l'intérieur d'une trame : à l'émission, on insère dans le corps de la trame un élément binaire 0 après avoir rencontré cinq éléments binaires consécutifs de valeur 1.

En réception, il faut supprimer l'élément binaire de valeur 0 après avoir rencontré cinq éléments binaires consécutifs de valeur 1. Un tel mécanisme (le *bit stuffing*) interdit l'émission de plus de cinq éléments binaires de valeur 1 dans le corps de la trame, puisque cette configuration est réservée à sa délimitation. Cette méthode permet la transmission de trames de longueur quelconque sans contraintes particulières.

Exemple

Prenons les données utiles suivantes : `0110 1111 1110 1001`. Précédées et suivies de fanions, elles seront réellement émises sous la forme : `01111110 0110 1111 10110 1001 01111110`. Dans cette séquence, les fanions sont soulignés et le bit inséré pour la transparence est en gras souligné.

Délimitation par transmission de la longueur du champ de données

Une autre méthode de délimitation consiste à indiquer, dans un champ particulier, le nombre d'octets utiles contenus dans la trame. Après une séquence de début de trame, un ou plusieurs octets indiquent sa longueur (l'emplacement de ce champ est fixé par rapport au début de trame), qui s'exprime généralement en octets ou en nombre de mots (de 16 ou 32 bits, par exemple).

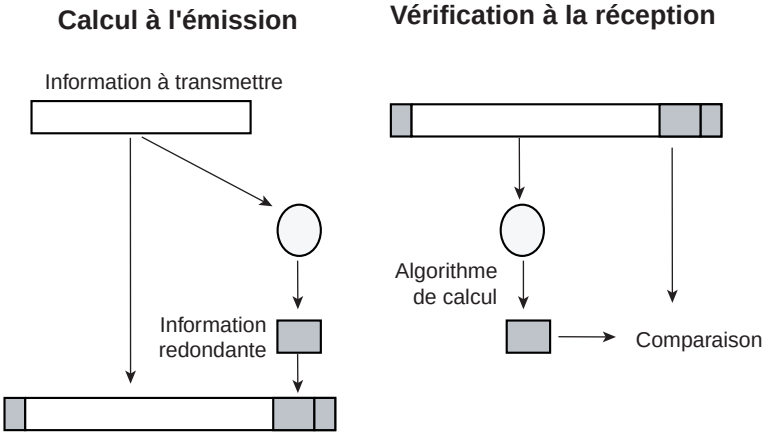
Ce procédé induit une limitation de la taille des trames : ainsi, si la longueur, exprimée en octets, est codée sur un octet, on se limite à des trames de 256 octets. Ce faisant, on évite les problèmes de transparence puisque le récepteur n'interprète en aucun cas les données reçues comme des délimiteurs. La longueur de la trame peut être ajustée à une longueur fixe par ajout d'éléments binaires de remplissage. Le champ précise alors la taille des données utiles transportées dans la trame.

1.2 Contrôle de la validité de l'information transmise

Le *contrôle d'erreurs* consiste à vérifier la validité des données transmises. Si on admet que le service de transmission n'est pas fiable, il faut se protéger contre d'éventuelles erreurs, donc les détecter puis les corriger. Pour cela, on ajoute à l'information transmise une *redondance*, c'est-à-dire des informations de contrôle calculées par un algorithme spécifié dans le protocole à partir du bloc de données. À la réception, on exécute le même algorithme pour vérifier si la redondance est cohérente. Si c'est le cas, on considère qu'il n'y a pas d'erreur de transmission et l'information reçue est traitée ; sinon, on est certain que l'information est invalide et elle est ignorée. La figure 2.1 illustre le principe de contrôle de validité de l'information transmise.

Figure 2.1

Principe de calcul du code de contrôle de validité.



La correction des erreurs se fait soit par l'intermédiaire d'une nouvelle tentative de transmission, soit en exploitant la richesse des informations de redondance, qui localisent et

corrigent les erreurs détectées. Lorsqu'on utilise une simple détection d'erreurs, le récepteur n'a aucun moyen de localiser l'erreur détectée : celle-ci peut se situer aussi bien dans le champ de redondance que dans le champ des données. Dans tous les cas, la trame est considérée comme invalide et ignorée du récepteur. Il est toujours possible que des erreurs de transmission apparaissent et que, par malchance, la cohérence reste vraie. On se trouve alors en présence d'*erreurs résiduelles*. Dans ce cas, le mécanisme de contrôle d'erreurs n'a pas pu détecter que plusieurs erreurs de transmission se sont mutuellement compensées. Le taux d'erreurs résiduelles doit être aussi faible que possible mais il ne peut jamais être nul sur une liaison de données réelle.

Remarque

Les erreurs résiduelles peuvent également exister sur des liaisons de données employant des codes correcteurs d'erreurs. Le nombre d'erreurs compensées a alors dépassé les capacités de correction du code correcteur, et le protocole de liaison ne peut pas signaler cet événement.

La détection des erreurs résiduelles ne s'effectue donc pas au niveau du protocole de liaison mais à des niveaux plus élevés de l'architecture de communication : par exemple, au niveau de l'application ou même seulement par l'utilisateur qui a reçu une information incohérente par rapport à ce qu'il attendait.

Contrôle de la validité : protection au niveau du code

La protection au niveau du code consiste à organiser une redondance interne à celui-ci : parmi toutes les combinaisons possibles, certaines sont retenues comme valides. Ce type de protection est possible lorsque l'émission des données se fait par caractère (on introduit une redondance pour chaque caractère transmis). Par exemple, on ajoute à chaque caractère un *bit de parité* dit *parité verticale* ou VRC (*Vertical Redundancy Check*)[1], calculé comme suit : pour chaque caractère, on fait la somme modulo 2 de ses bits. Si le nombre de bits 1 est pair, on ajoute 0 à la fin du caractère, et si le nombre de bits 1 est impair, on ajoute 1.

Le contrôle de validité par VRC est fréquemment utilisé avec le code CCITT n° 5 sur les liaisons asynchrones. Par exemple, pour le caractère *M* codé par `1001101`, le bit de parité vaut 0. On transmet dans cet ordre `10110010` (les 7 bits de données en commençant par les poids faibles puis le bit de parité). L'inconvénient général lié aux contrôles par parité est qu'on ne détecte pas les erreurs doubles.

Contrôle de la validité : protection au niveau de la trame

La protection au niveau des trames consiste à rajouter une redondance à chaque trame, en fonction de l'ensemble des éléments binaires qui la constituent. Plusieurs techniques sont envisageables, mais nous n'examinerons ici que la parité longitudinale et le contrôle polynomial, qui sont les méthodes les plus connues et les plus utilisées.

Contrôle de parité longitudinale ou LRC (*Longitudinal Redundancy Check*) Pour améliorer la détection des erreurs dans les transmissions utilisant les contrôles par parité, on associe souvent parité longitudinale et parité verticale (VRC + LRC). Pour cela on ajoute, à la fin de la trame, un mot de code appelé *parité longitudinale* ou LRC, constitué par la somme modulo 2 de tous les bits de même rang.

1. Par référence aux bandes magnétiques utilisées comme mémoire de masse. Sur la bande, vue comme un long ruban, on écrivait, d'une part, en parallèle les bits du caractère, d'où le nom de par té verticale donné à cette méthode de protection du caractère, et, d'autre part, les caractères les uns à la suite des autres, dans le sens longitudinal de la bande, pour constituer le bloc de données. Un autre code de protection était inscrit en fin de bloc.

Exemple

Soit la suite de caractères *L*, *2*, *M* à transmettre, codée en CCITT n° 5 par les valeurs hexadécimales *4C*, *32* et *4D*. En parité paire, les bits de parité (en gras dans le texte) pour chaque caractère valent respectivement 1, 1 et 0. Le caractère de parité longitudinale est calculé comme suit :

1 1 0 0 1 1 0 **0** caractère L + parité VRC ;

1 0 1 1 0 0 1 **0** caractère 2 + parité VRC ;

0 1 0 0 1 1 0 **1** caractère M + parité VRC ;

0 0 1 1 0 0 1 1 caractère du LRC à ajouter à la fin du bloc de données comme caractère de contrôle.

La suite des éléments binaires émise est donc 0011 0011 0100 1101 1011 0010 1100 1100, si on transmet les caractères les uns derrière les autres, en commençant par les poids faibles de chaque caractère.

Contrôle polynomial Le contrôle polynomial, appelé couramment par abus de langage *code cyclique* ou CRC (*Cyclic Redundancy Check*), est très utilisé dans les protocoles modernes car il permet de détecter les erreurs sur plusieurs bits. Nous nous contentons ici d'en décrire le processus sans en faire la théorie.

Dans le contrôle polynomial, on considère la trame à transmettre comme un groupe de bits auquel on fait correspondre un polynôme $P(x)$, tel que le coefficient de degré i correspond à la valeur du i^e bit. Les algorithmes de calcul se font modulo 2 sur les polynômes [par exemple, $(x^7 + x^3) + (x^3 + x) = x^7 + x$]. On choisit un polynôme $G(x)$ de degré r, appelé *polynôme générateur*, caractéristique du contrôle. À l'émission, on multiplie $P(x)$ par x^r et on divise le polynôme obtenu par $G(x)$. Le reste noté $R(x)$, obtenu par division euclidienne, est de degré strictement inférieur à r. Il est ajouté à la fin de la trame comme code de contrôle. Ainsi :

$$x^r * P(x) = G(x) * Q(x) + R(x). \quad (1)$$

On transmet le polynôme $T(x)$, constitué à partir de $P(x)$ et du reste $R(x)$ et défini par l'équation (2) :

$$T(x) = x^r * P(x) + R(x). \quad (2)$$

D'après les équations (1) et (2) et en tenant compte du fait que les calculs s'effectuent modulo 2, ce polynôme vérifie :

$$T(x) = G(x) * Q(x).$$

Il est donc divisible par G(x).

Nous avons vu que le circuit de données peut modifier l'information. Soit $E(x)$ le polynôme associé aux erreurs apportées par le circuit. Les données reçues ont pour polynôme associé $S(x)$, défini par : $S(x) = T(x) + E(x)$. À la réception, on divise $S(x)$ par $G(x)$ et on obtient un reste $R_1(x)$ qui vérifie l'équation suivante :

$$S(x) = G(x) * Q_1(x) + R_1(x).$$

Si $R_1(x)$ est nul, on considère que $E(x)$ est nul et que l'information reçue correspond à celle émise. Si $R_1(x)$ n'est pas nul, le polynôme $E(x)$ ne l'est pas non plus : le circuit de données a introduit une ou plusieurs erreurs et l'information reçue doit être ignorée.

À l'information 1000001110000100 est associée $P(x) = x^{15} + x^9 + x^8 + x^7 + x^2$.
Soit le polynôme générateur de degré 12 :

$$G(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1.$$

La division de $x^{12} * P(x)$ par $G(x)$ donne :

$$R(x) = x^{11} + x^9 + x^8 + x^7 + x^6 + x^4 + 1.$$

On transmet :

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

$x^{12}*P(x)$ $R(x)$

À la réception, on vérifie que le reste de la division par $G(x)$ est nul.

1.3 Modes d'exploitation d'une liaison de données

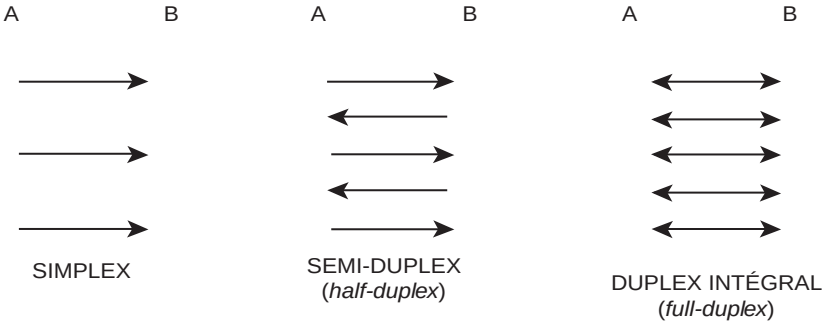
Le mode d'exploitation d'une liaison de données dépend du choix fait par le protocole de liaison (voir figure 2.2). La liaison peut être exploitée en *simplex*, *semi-duplex* (ou *half-duplex* ou *à l'alternat*), *duplex intégral* (ou *full-duplex*). Dans le mode simplex, l'échange de données se fait dans un seul sens. En semi-duplex, il se fait alternativement dans les deux sens, c'est-à-dire que les deux stations ne doivent pas transmettre simultanément (sinon il y a *contention*). En duplex intégral, les stations peuvent émettre simultanément sans aucune contrainte.

Remarque

Le mode d'exploitation de la liaison de données peut différer des caractéristiques du circuit de données : par exemple, on peut exploiter une liaison en semi-duplex alors que le circuit de données autorise des transmissions en duplex intégral.

Figure 2.2

Modes d'exploitation de la liaison.



2 Fonctionnalités d'un protocole de liaison

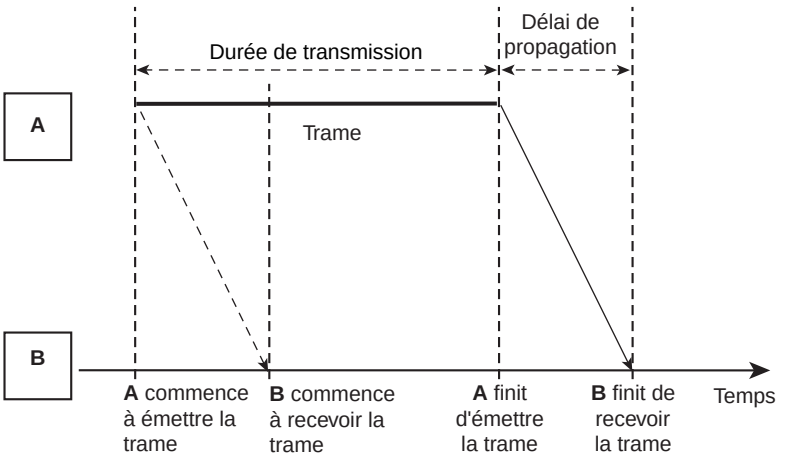
À partir d'un exemple simple de dialogue entre deux équipements, nous introduisons les différents types de protocoles de liaisons de données et les concepts qu'ils utilisent.

2.1 Représentation des échanges de données

Pour représenter les échanges de données, considérons que le temps s'écoule selon un axe horizontal. La transmission d'une trame est schématisée par un trait gras à la figure 2.3 et sa longueur représente la durée d'émission de la trame. Une flèche légèrement inclinée par rapport à la verticale représente la propagation de la trame ; la fin de la seconde flèche (côté destinataire) représente l'instant où la trame est totalement reçue.

Figure 2.3

Représentation des échanges.



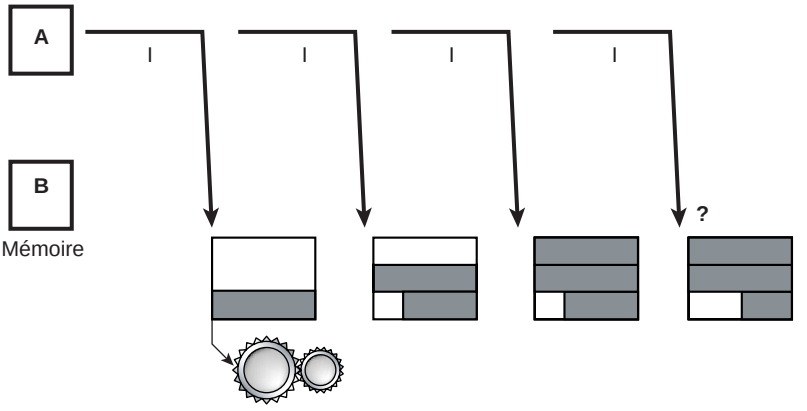
2.2 Contrôle de flux

Considérons deux équipements *A* et *B* reliés par un circuit de données sur lequel *A* veut envoyer des données à *B*. L'équipement *A* découpe les données en trames, appelées trames d'*information*, et les transmet les unes à la suite des autres. Elles sont repérées par la lettre *I*. Le circuit étant exempt d'anomalies, toutes les données sont délivrées sans erreur à l'équipement *B* qui les stocke pour les exploiter.

Supposons que *A* soit un ordinateur et *B* une imprimante lente, dotée d'une capacité mémoire limitée, lui imposant de garder en mémoire toutes les informations envoyées par *A* tant qu'elles ne sont pas imprimées. Si le rythme d'envoi des informations est nettement supérieur à son rythme d'impression, il y a rapidement saturation de la mémoire et perte d'informations par *B*. Il faut mettre en place un mécanisme de contrôle du rythme d'envoi des informations vers le récepteur, appelé *contrôle de flux*. Le petit engrenage de la figure 2.4 représente le mécanisme mis en jeu pour contrôler le flux des informations arrivant dans l'imprimante.

Figure 2.4

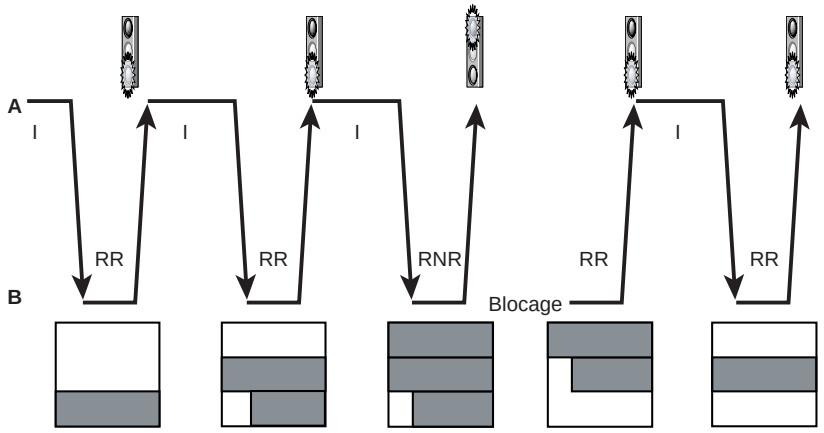
Exemple d'une transmission sans contrôle de flux sur un circuit de données parfait.



Pour réaliser le contrôle de flux, on introduit deux trames de *supervision*, RR (*Receiver Ready*) et RNR (*Receiver Not Ready*). Ces trames ne transportent aucune information utile et ne servent qu'à la gestion du dialogue. Elles sont générées et exploitées par le protocole de liaison et sont invisibles pour l'utilisateur. Le mécanisme est le suivant : à chaque réception de trame, l'équipement *B* envoie une trame RR s'il est prêt à accepter d'autres trames ou une

trame RNR s'il ne veut plus en recevoir de nouvelles. Dans ce dernier cas, *B* envoie RR dès qu'il est prêt à accepter de nouvelles trames, comme le montre la figure 2.5.

Figure 2.5
Mécanisme du contrôle de flux.



Il existe des variantes à ce mécanisme : par exemple, l'équipement *B* peut s'abstenir d'envoyer des trames RR. Lorsque la mémoire disponible descend au-dessous d'un certain seuil, l'équipement génère une ou plusieurs trames RNR et envoie des trames RR dès qu'une partie de la mémoire est libérée. Dans une autre variante, *B* peut transmettre en continu des trames RR tant qu'il a de la mémoire disponible (quelle que soit l'action de l'équipement *A*) et des trames RNR dès que sa mémoire est pleine. Un tel processus est couramment utilisé pour les liaisons entre ordinateur personnel et imprimante. Les trames de supervision sont alors réduites à deux caractères : la trame RNR est codée par le caractère *XOFF* (Ctrl+S), la trame RR par *XON* (Ctrl+Q).

2.3 Gestion des acquittements

Supposons maintenant que le circuit ne soit pas totalement fiable et introduise des erreurs. Au mécanisme de contrôle de flux décrit précédemment, il faut ajouter un processus d'*acquittement des trames d'information reçues.* Plusieurs options sont possibles :

- Lorsque l'équipement récepteur reçoit correctement une trame, il envoie une trame d'acquittement et ne fait rien en cas de mauvaise réception.
- Lorsqu'une trame est mal reçue, l'équipement récepteur envoie une *demande de retransmission* à l'émetteur et ne fait rien en cas de bonne réception.

Dans la seconde stratégie, l'absence de réponse est considérée comme un acquittement : à chaque trame, l'équipement émetteur arme un *temporisateur* correspondant à l'attente maximale d'une demande de retransmission provenant du récepteur ; si une telle demande parvient à l'émetteur, il répète la dernière trame. Dans le cas contraire, à expiration de la temporisation, l'émetteur considère que la transmission s'est bien effectuée. Cette stratégie présente deux inconvénients : elle est peu fiable car la demande de retransmission elle-même peut être mal transmise. Elle est en outre peu efficace puisqu'elle provoque une attente systématique en cas de bonne transmission.

Dans les protocoles de liaison de données, on utilise plutôt une stratégie d'acquittement positif, à l'aide des trames de supervision précédentes (RR et RNR). La stratégie de fonctionnement de *B* devient :

- Si *B* reçoit une trame correcte, l'équipement envoie un acquittement (trame RR ou RNR), selon l'état de sa mémoire pour assurer le contrôle de flux.

- S'il reçoit une trame erronée, il ne la mémorise pas et ne renvoie rien, comme s'il n'avait rien reçu.

Le principe de gestion des trames est le suivant : à l'émission de chaque trame d'information, *A* arme un temporisateur (correspondant à l'attente maximale de l'acquittement de *B*) qui sera désarmé à réception de l'acquittement correspondant. À l'échéance de cette temporisation, si *B* n'a pas répondu ou si sa réponse est brouillée, *A* réémet la trame et réitère le processus précédent. Le nombre de répétitions autorisées est limité : au-delà d'un certain seuil, on considère qu'un incident grave s'est produit (rupture totale de liaison, panne de l'équipement *B*, panne d'un élément de transmission sur *A* ou *B*…). Il faut alors avertir l'utilisateur que la liaison de données est rompue.

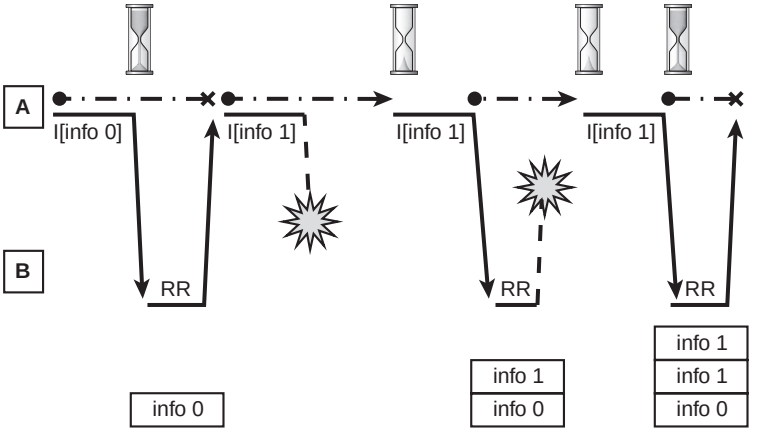
Ce mode d'acquittement n'est pas encore satisfaisant, car le circuit de transmission peut corrompre aussi bien les trames émises par *A* que celles émises par *B*. La figure 2.6 montre un échange de données mal conduit : une trame mal reçue par le récepteur se représente par un trait pointillé sans flèche ; les sabliers symbolisent les temporisations associées aux trames.

Figure 2.6

Protocole avec acquittement simple.

I[info 0] signifie que la trame I transporte l'information « info 0 ».

Dans cet exemple, la trame I'« info 1 » est dupliquée.



En effet, supposons que *A* envoie vers *B* une trame contenant l'information 1 et que *B* réponde à cette trame bien reçue par une trame RR qui est mal transmise (que l'équipement *A* ne reconnaît pas). Il réémet la même trame qui sera dupliquée dans *B*. Or, celui-ci n'a aucun moyen de détecter la duplication, puisqu'en aucun cas il n'analyse le contenu de la trame (*A* peut très bien décider de transférer deux fois de suite la même trame !). Puisque le protocole de liaison de données doit être *complètement indépendant du contenu* des trames transférées, il faut introduire un mécanisme supplémentaire qui distingue deux trames successives différentes. L'introduction d'un champ supplémentaire de *numérotation* ou d'*indication de retransmission* répond à ce besoin.

2.4 Numérotation des trames d'information

Le protocole numérote chaque trame d'information. La numérotation est placée dans l'en-tête, tout comme le type de la trame. Deux trames possédant des numéros différents sont considérées comme transportant des unités de données distinctes. Le protocole du récepteur exploite l'en-tête pour vérifier si la trame est correcte et en séquence. Dans l'affirmative, l'information contenue dans le champ de données est délivrée à l'utilisateur.

On appelle *N(S)* [*S* pour *send*] la variable donnant le numéro de la trame. Cette variable, codée sur quelques bits, est prise modulo *M*, un entier qui peut prendre les valeurs : 2 (le minimum pour distinguer deux trames successives différentes), 8 ou 128. L'introduction

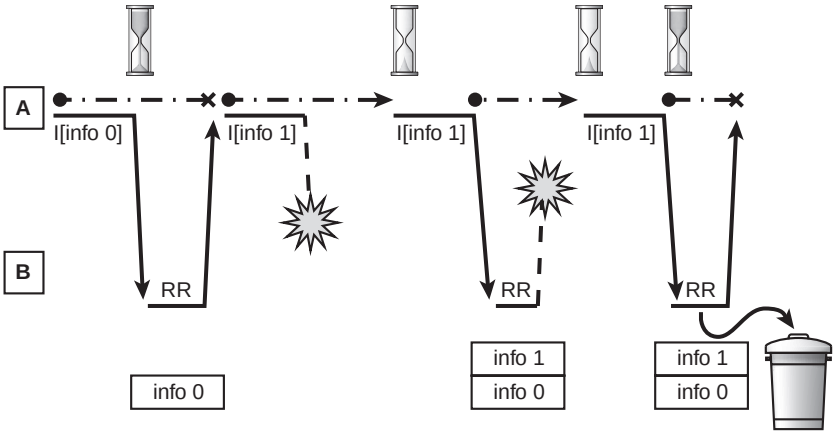
de numéros dans les trames impose des compteurs dans chaque station. De plus, il faut initialiser le dialogue pour que les deux stations se mettent d'accord sur les valeurs initiales des compteurs.

Le processus fonctionne de la façon suivante : *A* possède un compteur interne *V(S)* donnant le numéro de la prochaine trame à émettre. *A* émet cette trame en copiant *V(S)* dans le champ *N(S)*, puis il incrémente *V(S)*. Pour toute répétition d'une trame d'information, *A* émet la trame sans modifier son numéro d'ordre.

La figure 2.7 montre un échange de trames dont certaines sont erronées. La poubelle matérialise l'élimination des trames par le récepteur. Sur réception d'une trame, *B* teste la valeur du compteur *N(S)* : si sa valeur est égale à celle de la trame précédente, la trame est dupliquée et *B* l'ignore. Il peut cependant envoyer une trame de supervision pour acquitter la trame qu'il vient de recevoir.

Figure 2.7

Protocole à numérotation de trames.



Ce mécanisme possède une grosse faiblesse puisque la valeur de la temporisation est un paramètre critique. En effet, un mauvais choix de valeur peut entraîner un dysfonctionnement du protocole, comme le montre l'exemple de la figure 2.8.

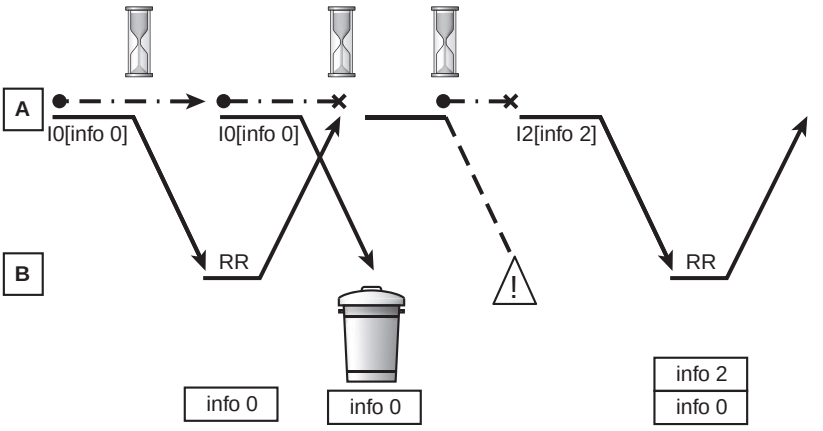
Considérons le scénario où *A* envoie une trame I_0 bien reçue par *B* mais dont l'acquittement lui arrive après expiration de la temporisation associée ; *A* émet à nouveau la trame I_0. Lorsqu'il reçoit l'acquittement de la première trame I_0 émise, il considère qu'il s'agit de l'acquittement de la seconde et il émet la trame I_1. *A* interprète l'acquittement qui suit comme l'acquittement de sa trame I_1. Si, par malheur, I_1 est mal transmise, elle va manquer à *B : A* croit qu'elle est acquittée alors qu'en réalité *B* ne l'a pas reçue. On est donc obligé de préciser dans une trame d'acquittement le numéro de la trame d'information qu'elle acquitte.

2.5 Notion de fenêtre

Afin d'augmenter l'efficacité du dialogue, on introduit la notion d'*anticipation,* c'est-à-dire la possibilité d'émettre plusieurs trames à la suite, sans avoir reçu l'acquittement des trames précédentes. Ainsi, une trame de supervision n'acquitte plus une seule trame mais *un ensemble de trames qui se suivent sans erreur.* Le nombre de trames successives qu'on peut émettre sans réception d'acquittement est limité par une valeur notée *k*, appelée *fenêtre*. Considérons une numérotation des trames modulo 8. Intuitivement, on perçoit que la valeur maximale de *k* est au plus égale à 8. Nous montrons qu'elle doit être au plus de 7.

Figure 2.8

Exemple de mauvais fonctionnement avec des trames numérotées et des acquittements sans numérotation.

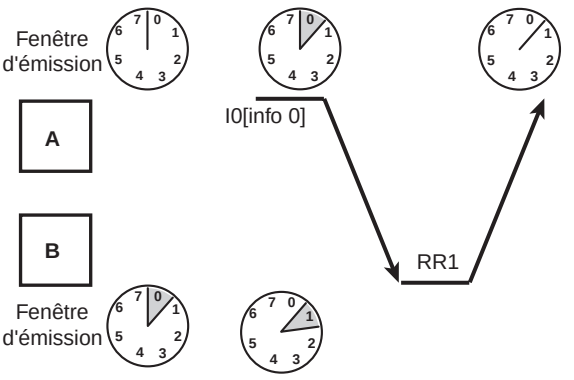


Adoptons une représentation circulaire des trames en attente d'acquittement. À la figure 2.9, un disque représente l'ensemble des valeurs des numéros d'une trame ; chaque trame occupe une part du disque. Un trait gras représente la valeur *V(S)* du compteur interne de *A* donnant le numéro de la prochaine trame à émettre.

À l'instant initial, *A* est au repos. Dès que *A* lance l'émission de la trame 0, cette trame est considérée comme émise mais non acquittée : on noircit la portion 0 du disque. Si *A* émet plusieurs trames successives, on noircit l'ensemble des trames en attente d'acquittement. Lorsque *A* reçoit un acquittement, les portions correspondant aux trames acquittées sont blanchies. L'ensemble des portions noircies représente l'état de la *fenêtre d'émission.* On représente également par un disque l'état du récepteur en noircissant les numéros que *B* s'attend à recevoir. À l'initialisation, *B* s'attend à recevoir la trame numérotée 0 : la case 0 est noircie. Lorsque cette trame est reçue correctement, *B* se met en attente de la trame 1 ; la case 1 est, par conséquent, noircie à son tour et ainsi de suite.

Figure 2.9

Représentation des fenêtres d'émission et de réception.



En absence d'erreur de transmission, le fonctionnement est le suivant : dès que *B* reçoit une trame, il enregistre son numéro *N(S)*, l'incrémente de 1, le mémorise dans une variable interne *V(R)* puis place cette valeur dans le champ *N(R)* de la trame de supervision qu'il renvoie à *A*. Tant qu'il en reste, l'équipement *A* émet ses trames, à moins qu'il n'ait atteint le nombre de trames autorisées sans réception d'acquittement. Une temporisation, armée à l'émission de chaque trame, est désarmée chaque fois que la trame correspondante est acquittée. On remarque que ce protocole suppose des équipements fonctionnant en mode duplex intégral car les acquittements sont reçus pendant l'émission des trames. La figure 2.10 illustre ce procédé.

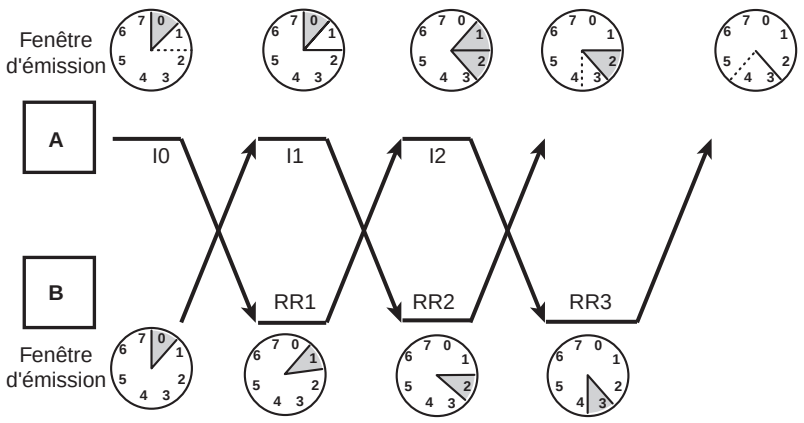


Figure 2.10

Scénario pour un protocole à fenêtre d'anticipation de largeur 2.

Lorsque A a émis les trames I_0 et I_1, sa fenêtre est fermée. Il attend la réception d'un acquittement pour pouvoir émettre la trame I_2.

Déterminons par un exemple la taille maximale de la fenêtre d'anticipation, lorsque les trames sont numérotées sur 3 bits (les valeurs vont de 0 à 7). Considérons les deux scénarios suivants :

- *A* transmet une trame numérotée 0, acquittée par *B*. L'acquittement n'est pas reçu par *A* qui émet à nouveau la trame 0 à expiration de la temporisation associée.
- *A* transmet une trame numérotée 0, acquittée par *B* à l'aide de la trame *RR1*. Ensuite, huit trames sont successivement émises (les trames 1 à 7 puis la trame 0), mais les sept premières ne sont pas reçues par *B*.

Dans le dernier scénario, de son point de vue, *B* a reçu deux trames successives portant l'indice 0, autrement dit, deux fois la même trame alors qu'il s'agit de deux trames différentes. Il est donc nécessaire de limiter la fenêtre d'anticipation à sept trames pour éviter toute confusion. De façon générale, si les trames sont numérotées de 0 à *n*, la taille maximale de la fenêtre d'anticipation est au plus *n* : graphiquement, il doit toujours y avoir une part du disque non noircie pour éviter toute ambiguïté dans l'acquittement.

Quand une trame reçue est erronée, elle n'est pas prise en compte. Une erreur ne sera alors détectée que si l'une des trames I suivantes est correctement reçue. Par contre, son numéro ne correspondra pas au *N(S)* attendu. Deux stratégies sont envisageables : on réémet toutes les trames à partir de la trame erronée (*Go-back-N*) ou on ne réémet que la trame erronée par un mécanisme de *rejet sélectif* (*Selective Reject*).

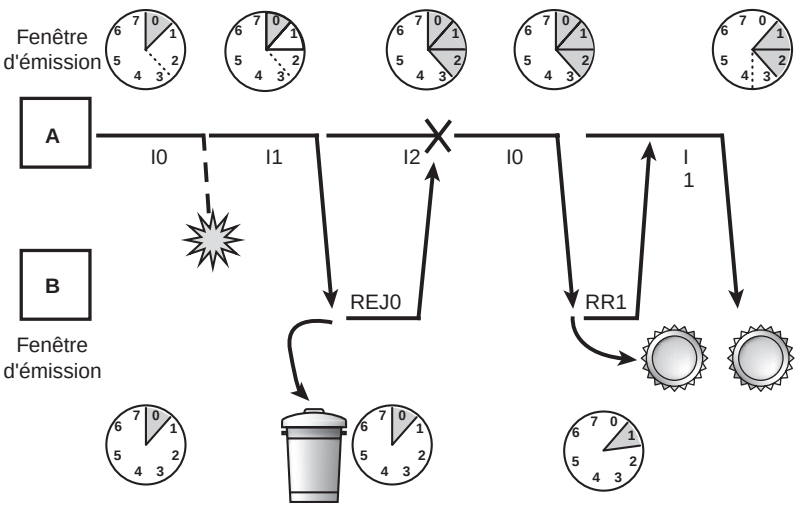
2.6 Protocole Go-back-N

Dans la stratégie Go-back-N (retour au *n-ième*), une trame de supervision appelée REJ (*Reject*) sollicite la retransmission des trames à partir de la trame erronée.

Le Go-back-N est illustré à la figure 2.11 et respecte le scénario suivant : *A* envoie la trame 0 (mal reçue), suivie de la trame 1 (bien reçue). En recevant la trame 1, *B* constate une rupture de séquence : il a reçu la trame 1 sans avoir reçu de trame 0. De ce fait, il ne mémorise pas la trame 1 et envoie une trame REJ avec le numéro 0 pour demander la reprise d'émission à partir de la trame 0. En recevant la trame REJ0, *A* interrompt éventuellement l'émission de la trame en cours pour reprendre le processus d'émission à partir de la trame erronée.

Figure 2.11

Scénario d'un protocole Go-back-N.



2.7 PIGGY-BACKING

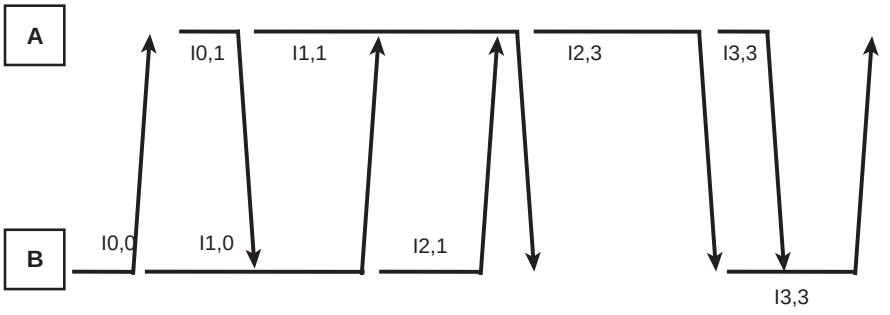
Dans des échanges bidirectionnels, chaque équipement envoie des trames d'information numérotées et acquitte les trames I qu'il a reçues (voir figure 2.12). Il y a donc deux sortes de trames émises : les trames I et les trames d'acquittement (RR, RNR ou REJ selon l'état de la réception). Un tel mécanisme n'est pas très efficace ; on l'améliore en utilisant les trames I pour véhiculer à la fois les informations à émettre et les acquittements des trames reçues précédemment. Chaque trame I possède deux numéros : un numéro *N(S)* [le numéro d'ordre de la trame I], et un numéro *N(R)* acquittant les trames émises dans le sens opposé. Ce mécanisme est appelé *piggy-backing*. Enfin, lorsqu'une station n'a pas de trame I à émettre, elle peut toujours utiliser des trames RR pour acquitter le trafic qu'elle reçoit.

Figure 2.12

Scénario d'un protocole duplex intégral avec piggy-backing.

Une trame I N(S),N(R) a comme signification I N(S) et RRN(R).

La fenêtre d'anticipation à l'émission est de taille 3 au moins dans cet exemple.



2.8 CONCLUSIONS

Un protocole de liaison de données peut offrir plusieurs services suivant la qualité de la transmission :

- *Service sans acquittement, ni connexion, ni contrôle de flux* lorsqu'on souhaite utiliser un protocole très simple ou lorsque le circuit de données est d'excellente qualité.
- *Service avec acquittement mais sans connexion, ni contrôle de flux* qui permet d'améliorer un peu la fiabilité de la liaison mais ne garantit pas la non-duplication des messages.
- *Service avec acquittement, connexion et contrôle de flux* qui inclut la numérotation des trames et des acquittements. Ce service, le seul à offrir une réelle garantie de fiabilité,

est aussi le plus complexe à implanter. On distingue plusieurs stratégies dans la gestion des acquittements : le *Stop-and-Wait* (utilisation d'une fenêtre d'anticipation égale à 1), le *Go-back-N* et le *Selective Reject*. Le *Stop-and-Wait* est peu efficace, le *Go-back-N* est le plus utilisé ; le *Selective Reject* n'apporte pas de gain flagrant de performances dans la majorité des cas.

De multiples protocoles de liaison de données ont été développés. Nous nous contentons ici de la présentation détaillée d'un seul protocole : HDLC (*High level Data Link Control*), une recommandation internationale datant des années 1970.

3 Description du protocole HDLC (*High level Data Link Control*)

HDLC est le protocole normalisé par l'ITU (*International Telecommunications Union*[2]), qui décrit une transmission en duplex intégral fonctionnant sur une liaison point à point ; la transmission est synchrone et orientée bit. Ce protocole met en œuvre le mécanisme de transparence décrit en début de chapitre, ce qui le rend totalement indépendant du codage des données transportées. HDLC peut transporter des informations utilisant des codes de longueur variable. Sa variante la plus connue est de type Go-back-N avec un mécanisme de contrôle de flux. Il fonctionne en mode équilibré ou symétrique[3], c'est-à-dire que les deux stations ont les mêmes prérogatives et peuvent éventuellement fonctionner selon un mode half-duplex.

3.1 Structure d'une trame HDLC

La trame est la structure unique de longueur quelconque qui transporte toutes les informations. Un fanion en marque le début et la fin ; un seul fanion marque la fin d'une trame et le début de la suivante lorsque deux trames sont émises consécutivement. Le tableau 2.1 décrit les différents champs de la trame, dans leur ordre d'apparition :

- Le champ *Address* s'étend sur un octet et identifie une des extrémités de la liaison.
- Le champ *Control* décrit le type de la trame : il s'étend sur 1 octet (sur 2 octets dans le *mode étendu*).
- Le champ *Information* est facultatif. Il contient un nombre quelconque d'éléments binaires représentant les données de l'utilisateur.
- Le champ FCS (*Frame Control Sequence*) est la séquence de contrôle de trame, obtenue par un contrôle polynomial dont le polynôme générateur vaut $x^{16} + x^{12} + x^5 + 1$. Ce polynôme générateur est celui préconisé par la recommandation *V41* de l'ITU.

Tableau 2.1

Format de base des trames HDLC

Flag	Address	Control	Information	FCS	Flag
01111110	8 bits	8 bits	*N* bits	16 bits	01111110

Le champ de gauche est le premier transmis, le champ de droite est le dernier.

2. On trouve également le sigle français UIT (*Union internationale des télécommunications*) pour désigner la même instance internationale. Nous utiliserons le sigle anglais.

3. Dans ce mode, chaque équipement possède deux fonctions : une primaire, qui émet des requêtes, et une secondaire, qui envoie des réponses aux requêtes.

On commence par émettre les bits de poids faibles (du bit 1 au bit 8 de chaque champ). La transmission d'éléments binaires est continue ; en l'absence d'émission spécifique, les équipements émettent des suites de fanions pour maintenir la synchronisation entre les deux extrémités de la liaison de données.

3.2 Différents types de trames HDLC

Il existe trois types de trames identifiés par le champ Control : les trames d'information ou trames *I* permettent la transmission de données de l'utilisateur. Les trames de supervision ou trames *S* permettent l'acquittement et le contrôle de flux ; elles ne transportent pas de données, de même que les trames non numérotées ou trames *U* (*Unnumbered*)[4]. Ces dernières servent à commander la liaison : initialisation, libération, notification d'erreurs irrécupérables… Seule une trame *I* peut transmettre des données ; elle est numérotée par la variable *N(S)* et contient également l'acquittement des trames reçues en sens inverse (procédé de piggy-backing), grâce au numéro *N(R)*. Le tableau 2.2 montre le format du champ Control d'une trame I.

Tableau 2.2

Format de l'octet Control des trames I

8	7	6	5	4	3	2	1	
N(R)			P/F	*N*(S)			**0**	Format général des trames *I*

Le bit 1 de valeur 0 est spécifique à la trame I. La valeur du bit P/F dépend du statut de l'équipement (primaire ou secondaire) et de la nature de la trame (requête ou réponse[5]).

Trames de supervision (trames *S*)

Les trames *S* acquittent les trames *I* et indiquent l'état de disponibilité des stations (aptitude ou non à recevoir de nouvelles trames *I*). Contenant un numéro *N(R)*, elles servent au contrôle d'erreurs et au contrôle de flux. Les trois trames de supervision[6] sont :

- La trame RR indique que l'équipement est prêt à recevoir de nouvelles trames *I*. Le numéro *N(R)* donne le numéro de la prochaine trame attendue. Il signifie que *toutes* les trames *I* de numéro *N(S)* strictement inférieur à *N(R)* ont été reçues. Un équipement peut aussi envoyer des trames *RR* pour indiquer son état ou pour demander l'état de la station située à l'autre extrémité.
- La trame RNR acquitte les trames reçues et indique en outre que l'équipement n'est pas en mesure de recevoir de nouvelles trames *I*. Le numéro *N(R)* a la même signification que dans la trame *RR*.
- La trame REJ sert à demander l'arrêt immédiat des émissions en cours et une retransmission à partir de la trame *I* portant le numéro indiqué dans *N(R)*.

Le tableau 2.3 donne le format de l'octet Control des trames *S*.

Tableau 2.3

Format de l'octet Control pour les trames S de supervision

8	7	6	5	4	3	2	1	Format général des trames *S*
N(R)			P/F	0	0	**0**	**1**	*RR*
N(R)			P/F	0	1	**0**	**1**	*RNR*
N(R)			P/F	1	0	**0**	**1**	*REJ*

4. Nous verrons plus loin dans le protocole PPP qu'il existe des trames U, appelées trames UI (*Unumbered Information*), qui transportent des données de l'utilisateur.

5. L'interprétation du bit 5 (bit *P* ou *F*) dépend du statut de l'équipement. Une trame de requête est émise par la fonction primaire qui gère le bit *P*. Une trame de réponse, émise par la fonction secondaire de l'autre équipement, utilise le bit *F*.

6. Une quatrième trame de supervision, la trame *SREJ*, a été définie pour le rejet sélectif, mais ce mode de rejet des trames erronées ne fait pas partie de la version normalisée décrite ici. Nous verrons son fonctionnement au cours des exercices.

Trames non numérotées (trames *U*)

On utilise les trames *U* pour les fonctions supplémentaires de commande de la liaison. Citons les principales :

- SABM (*Set Asynchronous Balanced Mode*) pour initialiser le fonctionnement en mode équilibré.
- DISC (*DISConnect*) pour rompre logiquement la liaison entre les deux stations.
- UA (*Unnumbered Acknowledgement*) pour acquitter des commandes comme SABM ou DISC.
- FRMR (*FRaMe Reject*) pour rejeter une commande invalide (correcte du point de vue de la détection des erreurs mais incohérente par rapport à l'état du dialogue).
- DM (*Disconnect Mode*) pour indiquer l'état de déconnexion d'une station. Elle s'utilise, en particulier, pour répondre négativement à une demande d'initialisation par SABM.

Le tableau 2.4 donne le format de l'octet Control des trames *U*.

Tableau 2.4

Format de l'octet Control pour les trames U

8	7	6	5	4	3	2	1	
0	0	1	P/F	1	1	**1**	**1**	*SABM*
0	1	0	P/F	0	0	**1**	**1**	*DISC*
0	1	1	P/F	0	0	**1**	**1**	*UA*
1	0	0	P/F	0	1	**1**	**1**	*FRMR*
0	0	0	P/F	1	1	**1**	**1**	*DM*

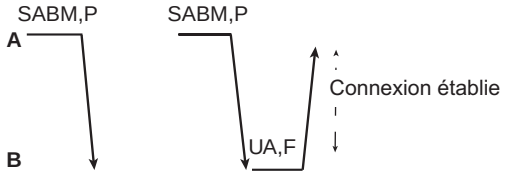
3.3 Établissement et libération de la liaison de données

HDLC est un protocole *orienté connexion* : il faut établir la liaison avant d'envoyer des informations ou en recevoir. De même, lorsque l'un des équipements veut terminer le dialogue, il engage une *procédure de libération*. La figure 2.13 décrit un problème survenu lors d'une tentative de connexion.

Figure 2.13

Initialisation de la liaison de données.

A tente d'établir une connexion vers B qui ne détecte pas la première trame mais acquitte la seconde.



N'importe quel équipement peut initialiser la liaison. Le primaire de l'équipement initiateur envoie la trame SABM,P et attend la réponse du secondaire de l'autre équipement. En cas de non-réponse, il réitère son envoi jusqu'au nombre maximal de tentatives de connexion. Au bout de ce nombre d'essais infructueux, il considère que la liaison est impossible à établir. À réception d'une SABM,P, le récepteur transmet une trame UA,F si son utilisateur accepte le dialogue, sinon il envoie une trame DM,F. Dans le premier cas, la connexion est alors établie ; tous les compteurs et les temporisateurs sont initialisés. Les premières trames émises de chaque côté porteront un *N(S)* égal à 0. Dans le second cas, les équipements entament une phase de libération. Ce dernier processus, symétrique à l'établissement de la liaison, utilise les commandes DISC et UA.

4 Cas particulier du protocole PPP (*Point to Point Protocol*)

Le protocole PPP est le protocole de liaison point à point utilisé dans Internet. Il utilise les lignes téléphoniques de l'abonné pour accéder au réseau (la liaison concerne typiquement un ordinateur personnel et le fournisseur d'accès à Internet). Il s'agit d'une version très simplifiée d'HDLC qui ne comprend – sauf options – ni contrôle de flux, ni mécanisme de reprise sur erreurs. La figure 2.14 donne la structure d'une trame PPP.

Figure 2.14
Format d'une trame PPP.

Les 8 bits du champ Address sont à 1 (la liaison étant point à point, une seule valeur d'adresse suffit).

Le champ Control a la même signification que dans HDLC.

Le champ Data PPP commence par deux octets (le champ protocole), qui identifient le protocole de niveau supérieur auquel est destinée la trame ; il se termine par un champ FCS dont le mode de calcul est identique à celui d'une trame HDLC.

La seule trame transportant des données sur une liaison fiable est une trame *U* de type *UI* (*Unnumbered Information*). Cette trame contient un champ d'informations mais n'est pas numérotée (car il n'y a pas de contrôle de flux). L'absence de mécanisme de reprise sur erreur ne signifie pas que le circuit est fiable : le champ FCS sert à valider les trames reçues.

PPP comprend également un ensemble de sous-protocoles choisis à l'ouverture de la liaison de données pour sécuriser les échanges : LCP (*Line Control Protocol*), PAP (*PPP Authentification Protocol*), CHAP (*Challenge Authentification Protocol*) et NCP (*Network Control Protocol*).

À l'initialisation d'un transfert, PPP négocie les paramètres de l'échange par le protocole LCP *;* PAP autorise l'échange – en clair – des mots de passe avant le transfert des données. Si on souhaite un échange sécurisé, on peut utiliser CHAP, qui effectue un chiffrement tout au long de la communication grâce à un échange préalable de clés publiques et de clés secrètes (voir les compléments pédagogiques sur le site www.pearsoneducation.fr). Enfin, le protocole NCP sert à négocier les paramètres de connexion (les options de transfert choisies par chaque extrémité de liaison, indépendamment l'une de l'autre) et les paramètres de niveau réseau (par exemple, l'affectation des adresses IP, que nous verrons au chapitre 6).

Résumé

Le protocole de liaison de données supervise le circuit de données et définit un ensemble de règles pour assurer la fiabilité des échanges. Il spécifie le format des trames, les moyens de contrôler leur validité, ainsi que les règles du dialogue entre les deux extrémités de la liaison. Il exerce aussi un contrôle de flux pour maîtriser le rythme d'envoi des informations et valider la réception des informations reçues.

HDLC est un exemple de protocole normalisé très répandu, qui gère des trames de données à l'aide de trames de supervision. Il fonctionne en full-duplex et permet la reprise sur erreur. Il garantit en outre l'ordre des données. PPP, utilisé dans Internet, en est une version très simplifiée qui n'exerce pas de contrôle de flux mais propose plusieurs modes d'échanges de données, négociées avant tout transfert de données entre les deux extrémités de la liaison.