Chapitre 03 : La couche liaison de données

1. Introduction

La couche physique assure le transfère de bits dans un canal de transmission. Cependant, la transmission de données n'est pas un processus fiable. En fait, les bits envoyés peuvent être se subis aux plusieurs problème (comme les erreurs ou les pertes). Ainsi, la couche liaison de données a pour but de fiabiliser la transmission.

2. Fonctions de couche liaison de données

Le but de la couche de liaison de données est de faire communiquer deux machines adjacentes, c'est-à-dire deux machines physiquement liées par un canal de transmission. Naturellement, cette couche s'appuie sur la couche physique et fournit des services à la couche réseau. En fait, la couche liaison offre trois services à la couche réseau : le service sans connexion et sans accusé de réception, le service sans connexion avec accusé de réception et le service orienté connexion. Bien entendu, la couche réseau utilise ces service en appelant certaines primitives comme : ***Request***, ***Indication***, ***Response*** et ***Confirmation***.

Les protocoles de liaison de données travaillent sur des blocs de bits appelés ***trames***. Ces protocoles doivent, donc, définir la structure syntaxique des unités de données.

La couche liaison permet aussi de réguler le flux des données en adaptant la vitesse de l'émetteur à celle de récepteur afin d'éviter la perte de certaines trames. En fait, la charge éventuelle de récepteur l'empêche de traiter toutes les trames émises. D'autre part, il est impossible pratiquement d'équiper le récepteur par des mémoires illimitées.

Sachant que la transmission sur n'importe quel support ne jamais être parfaite, les protocoles de liaison doivent assurer la gestion de ces erreurs. Par gestion d'erreurs nous entendons ici la détection et la correction, si possible, d'erreurs.

3. Structure de trame

Les bits transmis par la couche physique peuvent comporter des erreurs. Afin de détecter (et corriger) ces erreurs, ces flux de bits doivent être découpés en trame et en ajoutant pour

chaque trame un somme de contrôle. Une trame est l'unité de données du protocole de niveau Liaison de données (L-PDU). Le découpage en trames peut être effectué selon plusieurs techniques:

- Insertion des silences pour délimiter les trames : cette technique consiste à faire des périodes d'attente (de repos) entre l'envoie de chaque deux trames consécutives. En conséquence, des silences seront apparaître entre les trames. Cependant, cette technique porte le risque de disparition de silence entre ces trames, ou l'apparition de silence au sein d'une trame.
- Compter les caractères : cette méthode consiste à déclarer au début d'une trame le nombre de caractère de cette dernière. Cependant, si le bruit affecte ce champ, on perdra la synchronisation avec l'impossibilité de resynchroniser ultérieurement.
- Utilisation des fanions de signalisation avec remplissage d'octets : ici on utilise un caractère spécial appelé *Fanion* (ou flag) au début et à la fin de trame. Naturellement, ce caractère ne doit pas être apparu dans la séquence de données. Ainsi, si ce caractère appartient à l'information, on ajoutera un autre caractère spécial appelé *ESC* avant lui. De même façon on peut traiter l'apparition de ESC dans l'information (c'est-à-dire en ajoutant des caractères *ESC*). Ce mécanisme est appelé : la *transparence*.
- Utilisation des indicateurs de début et de fin de trame avec le remplissage de bits : chaque trame commence et finit par la séquence binaire particulière suivante 01111110. Le mécanisme de transparence dans cette technique consiste à ajouter un bit 0 après chaque cinq bits 1 consécutifs appartient à l'information.
- La violation du codage physique : dans certaines techniques de codage physique il y a des codes non utilisables. Par exemple dans la technique de codage Manchester, l'absence de transition durant un intervalle significatif n'est pas un code admet. En conséquence, nous utilisons ces codes pour délimiter la trame.

4. Détection et correction d'erreurs

En transmettent un bloc de bits, on peut trouver des bits qui changent leurs valeurs. Le but de la couche liaison est de détecter (et éventuellement corriger) ce changement. Il existe plusieurs techniques qui servent à la détection et la correction des erreurs. Cependant, toutes

les techniques ont un seul principe qui consiste à l'ajout d'une suite de bits à l'information de base. On appel cette séquence ajoutée des bits de contrôle ou de redondance. L'ajout des bits de contrôle rend certains mots invalides. En conséquence, il y a une forte possibilité de tomber dans un mot invalide si une erreur change la valeur de certains bits. Si on peut reconnaitre l'invalidité de mot sans pouvoir reconnaitre le mot initial on dit que la technique est de détection d'erreur. Si on peut reconnaitre le mot initial à partir de mot invalide, on dit que la technique est correctrice d'erreur.

Le rendement d'un code est le taux de chiffre mot initial par rapport au mot à transmettre. Si un code (k, n) transforme un boc de *k* bits en *n* bits en ajoutant (n – k) bits de redondance, le rendement sera, donc, R = k / n.

4.1. La distance de Hamming

Soient deux mots M et M' de même longueur. On appelle la distance de Hamming $d_H(M, M')$, le nombre de symboles (c.-à-d. de positions) pour lesquels *M* et *M'* diffèrent. Formellement, $d_H(M, M') = |i \in \{1, .. n\} / M_i \neq M'_i|$

Pour un codage donné, la distance de Hamming désigne la distance minimale entre qui les mots que les constituent.

La distance de Hamming permet de caractériser le nombre d'erreurs que peut détecter ou corriger un code. Ainsi, si la distance minimale d'un code soit *d* alors on peut détecter au maximum *N* erreurs tel que $N = d - 1$. Cependant, la correction de *N* erreurs nécessite au moins une distance minimale *d* tel que $d = 2N + 1$.

4.2. Méthodes pour la détection d'erreurs

La méthode la plus simple de détection d'erreurs est la méthode appelée de parité paire (respectivement impaire) où on ajoute un bit sachant que le nombre de tous les bits soit paire (respectivement impaire). Considérant la technique de parité paire, le bit de contrôle *C* est calculé en utilisant la formule suivante: $C = \bigoplus_{i=1}^{n} b_{i,}$, tel que b_i est le bit dans la position *i* et n est la taille de message initial.

A la réception de message, on compte le nombre des 1 : si on le trouve paire (respectivement impaire) alors aucune erreur détectée, sinon le message transmis est

erroné. Prenant le cas de parité paire, le récepteur va calculer la somme $S = C \oplus (\oplus_{i=1}^{n} b_{i,})$; si S égale à zéro donc aucun erreur détectée sinon le message reçu est erroné. Cette méthode ne permet que la détection des erreurs dont le nombre est impaire.

Pratiquement dans les réseaux, la méthode la plus utilisée est la méthode appelée le code polynomiale ou CRC (pour ***Cyclic Redundancy Chek***). Cette méthode est basée sur le traitement des séquences binaires comme des polynômes binaires. Ainsi, la séquence binaire représente les coefficients d'un polynôme. A titre d'exemple, la suite binaire suivante : 1001011 peut être décrite par le polynôme suivant :

$1* x^6 + 0* x^5 + 0* x^4 + 1* x^3 + 0* x^2 + 1* x^1 + 1* x^0 = x^6 + x^3 + x^1 + 1.$

Afin d'utiliser cette méthode l'émetteur et le récepteur doivent se mettre d'accord sur le choix d'un polynôme générateur G(x). L'émetteur doit calculer un total de contrôle (*checksum*) et l'envoyer avec le message. Le récepteur utilise le même polynôme générateur pour vérifier si le code est valide ou non. On note, que les opérations effectuées par l'émetteur et le récepteur sont basées sur l'arithmétique modulo 2.

Le calcul du total de contrôle se fait à l'aide de l'algorithme suivant :

- Soit *r* le degré de *G(x)*. Ajouter *r* bits zéro après le bit de poids faible du bloc. On trouve ainsi un polynôme $x^rM(x)$.
- Effectuer la division ***modulo 2*** du polynôme $x^rM(x)$ par *G(x)*.
- Soustraire ***modulo 2*** le reste de la division de la chaine de bits correspondant au polynôme $x^rM(x)$. Le résultat est la trame transmise au destinataire (appelée *T(x)*)

Prenant l'exemple suivant : on veut utiliser la technique CRC pour la transmission de message 1001001101 avec un polynôme générateur x^4+x^1+1. Le polynôme générateur $G(x) = x^4+x^1+1$ peut être écrit en binaire par la suite : 10011. Dans la suite de ce paragraphe nous appliquons la technique CRC :

- Le degré de G(x) est 4, donc on ajoute 4 zéro à la fin de la séquence initiale M = 1001001101. On obtient donc la séquence M' = 1001001101 **0000.**

- Nous effectuons la division ***modulo 2*** de la séquence ***M'*** par ***G(x)*** comme il est montré au-dessous.

```
10010011010000 | 10011
10011          | 100010100
   10110
   10011
      10110
      10011
         10100
         10011
            1110
```

- On soustraire le reste de division (1110) de la séquence M', on obtient la séquence à envoyer soit : 10010011101**1110.**

A la réception d'un message ***M***, le récepteur va le diviser (***division modulo 2***) par le polynôme générateur **G(x).** Si le reste de division est zéro, donc le message reçu est sans erreur ; sinon le message est subi aux erreurs.

On note que le choix de polynôme doive être prudent afin de détecter toutes les erreurs.

4.3. Méthodes de correction d'erreurs

La technique polynômiale est utilisée pour la détection des erreurs. En effet, elle ne permet pas de reconstituer le message émis si des erreurs ont apparus. Cependant, il existe autres techniques qui permettent la reconstitution de message émis même des erreurs (en nombre limité) ont altéré le message.

Il est possible d'utiliser la technique de parité pour la correction d'erreurs. La technique de parité croisée consiste non pas à contrôler l'intégrité des données d'un caractère, mais à contrôler l'intégrité des bits de parité d'un bloc de caractères. En fait, nous utilisons un double contrôle de parité : le premier est *transversal* pour la détection l'erreur sur une ligne et le deuxième contrôle qui est *longitudinal* permet la détection d'erreur sur chaque colonne. Par exemple, on veut transmettre le message « *BONJOUR* » sur des liaisons. Chaque caractère est codé sur 07 bit en utilisant le code ASCII. Le 08ème bit est

le bit de contrôle transversal. En plus, on ajoute un autre caractère de contrôle longitudinal. Supposant que nous appliquons la technique de parité impaire, le tableau suivant représente le codage de chaque caractère avec les bits de contrôle.

Tableau 01 : codage d'un message par la technique parité croisée.

Les caractères	**Le code ASCII de caractère**							**Les bits de contrôle transversal**
B (1000010)	1	0	0	0	0	1	0	**1**
O (1001111)	1	0	0	1	1	1	1	**0**
N (1001110)	1	0	0	1	1	1	0	**1**
J (1001010)	1	0	0	1	0	1	0	**0**
O (1001111)	1	0	0	1	1	1	1	**0**
U (1010101)	1	0	1	0	1	0	1	**1**
R (1010010)	1	0	1	0	0	1	0	**0**
Les bits de contrôle longitudinal	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**0**

A la réception de message, le récepteur va calculer les bits de contrôle transversal et longitudinal. En calculant les bits de contrôle (transversal et longitudinal), on peut localiser le bit erroné qui représente l'intersection de ligne et colonne fausses.

Tableau 02 : La localisation d'une par la technique parité croisée.

Les caractères	**Le code ASCII de caractère**							**Les bits de contrôle transversal**
B (1000010)	1	0	0	0	0	1	0	**1** ***(OK)***
O (1001111)	1	0	0	1	1	1	1	**0** ***(OK)***
N (1001110)	1	0	0	0	1	1	0	**1** ***(Faux)***
J (1001010)	1	0	0	1	0	1	0	**0** ***(OK)***
O (1001111)	1	0	0	1	1	1	1	**0** ***(OK)***
U (1010101)	1	0	1	0	1	0	1	**1** ***(OK)***
R (1010010)	1	0	1	0	0	1	0	**0** ***(OK)***
Les bits de contrôle longitudinal	**0** ***(OK)***	**1** ***(OK)***	**1** ***(OK)***	**1** ***(Faux)***	**1** ***(OK)***	**1** ***(OK)***	**0** ***(OK)***	**0** ***(OK)***

Une autre technique basée sur la technique de parité est utilisée pour la correction des erreurs. Cette technique appelée le code de Hamming permet dans sa version la plus

simple de corriger un bit en erreur. Le principe de cette méthode consiste à ajouter des bites de correction (x_i) aux positions d'indice 2^i. Ainsi, pour chaque bit d'information il y a un ensemble de bits de correction qui contrôle sa parité. Par exemple, si on a une information codé sur 04 bits, alors il faut 03 bits de contrôle. Le tableau suivant donne pour chaque bit d'information, les bits de contrôle qui va vérifier sa parité.

Tableau 03 : La signification des bits de contrôle pour un message de 07 bits.

L'indice *i* de bit	Représentation binaire de *i*	Signification
7 = 4 + 2 +1	0111	Contrôlé par x_0 et x_1 et x_2
6 = 4 + 2	0110	Contrôlé par x_1 et x_2
5 = 4 + 1	0101	Contrôlé par x_0 et x_2
4	0100	Bit de contrôle (x_2)
3 = 2 + 1	0011	Contrôlé par x_0 et x_1
2	0010	Bit de contrôle (x_1)
1	0001	Bit de contrôle (x_0)

Les valeurs de ces bits sont calculées en utilisant une matrice H appelée la matrice de parité. Cette matrice est constitué d'un nombre de ligne égale au nombre de bits de contrôle et un nombre de colonne égale à la taille totale de message (information et bits de contrôle). Chaque colonne (*i*) est représentée en binaire. Le message est représenté par un vecteur A (avec des bits x_i de contrôle). En effet, en résolvant le système linaire $H*A^t = 0$, on peut obtenir les valeurs x_i.

Prenant l'exemple d'une information codé sur 04 bits (1101), en ajoutant les bits de contrôle, nous obtenons la séquence suivante : x_0 x_1 1 x_2 101.

Pour savoir les valeurs de x_i, nous résolvant le système linaire suivant :

$$\begin{matrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix} * \begin{matrix} x_0 \\ x_1 \\ 1 \\ x_2 \\ 1 \\ 0 \end{matrix} = \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$$

$$\begin{matrix} & & & & & & & & 1 \\ x_0 & & 1 & & & & & & \\ x_1 & = & 0 & & & & & & \\ x_2 & & 0 & & & & & & \end{matrix}$$

Le message à envoyer sera donc : 1010101.

Après la réception de message, le récepteur le multiple par la matrice de parité. S'il le résultat de multiplication est le vecteur nul donc le message reçu est sans erreur sinon le vecteur trouvé représente le code binaire de l'indice de bit erroné. Par exemple si le 5ème bit a subi à une erreur, donc le récepteur va recevoir 1010001. En multipliant le message par la matrice de parité, on va trouver :

$$\begin{matrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & & 1 & & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & * & 0 & = & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & & 1 & & 1 \\ & & & & & & & & 0 & & \\ & & & & & & & & 0 & & \\ & & & & & & & & 0 & & \\ & & & & & & & & 1 & & \end{matrix}$$

Le vecteur trouve est 101 = 5 qui représente le bit erroné.

5. Protocoles de liaison de données élémentaires

Dans cette partie on va présenter quelques protocoles de liaison de données par ordre de complexité croissante. Les protocoles présentés dans cette section utilisent un ensemble de fonctions et variables communs.

5.1. Protocole simplex non restreint

Ce protocole est irréaliste à cause de nombreuses hypothèses qui conditionnent son fonctionnement. Comme son nom indique, la transmission dans ce protocole est simplex. On suppose que les couches réseaux coté récepteur et émetteur soient toujours prêtes à émettre ou recevoir des paquets. Le canal de communication est parfait : aucune trame n'est erronée ni perdue.

Dans ce protocole il n y a pas de contrôle de flux, ni d'erreurs. L'émetteur prend les paquets de la couche réseau, les met dans des trames les envoie les unes après les autres. Le récepteur ne fait qu'attendre les trames et à leur arrivée ne fait que délivrer le paquet d'information à la couche réseau. Le protocole est présenté dans la figure 2.

```
type TypeEvenement = (ArriveeTrame); /* un seul type d'evenement */
procedure emetteur1;
var s: trame; /* tampon pour trame en emission */
tampon: paquet;
begin
  repeat
    Origine_Couche_Reseau(tampon); /* prendre ce qu'il y a à envoyer */
    s.info := tampon; /* on le copie dans s pour le transmettre */
    Vers_Couche_Physique(s);
  until FinDesTemps;
end; /*emetteur1*/

procedure recepteur1;
var r: trame;
evenement: TypeEvenement;
begin
  repeat
    attendre(evenement); /*Le seul evenement possible : ArriveeTrame*/
    Origine_Couche_Physique(r); /* acquisition de la trame */
    Vers_Couche_Reseau(r.info); /* on passe les donnees à la couche
reseau */
  until FinDesTemps;
end; /* recepteur1 */
```

Figure 2 : protocole simplex non restreint

5.2. **Protocole simplex de type arrêt et attente**

Dans ce protocole on va lever la deuxième hypothèse. La couche réseaux de destinataire n'est plus capable de traiter les données qui arrivent. L'émetteur doit émettre les trames selon les capacités de traitement du récepteur. Il attend, donc, jusqu'à l'arriver d'un acquittement qui l'informe sur l'état de récepteur. Le protocole est présenté dans la figure 3.

```
type TypeEvenement = (ArriveeTrame); /* Un seul type d'evenement */
procedure emetteur2;
var s: trame;
tampon : paquet;
```

```
evenement : TypeEvenement;
begin
repeat
Origine_Couche_Reseau(tampon) /* Acquisition du paquet venant de la
couche reseau*/
s.info := tampon; /* copie dans s pour transmission */
Vers_Couche_Physique(s); /* on envoie la trame */
Attendre(evenement); /* On attend un signal */
until FinDesTemps;
end; /*emetteur2*/
procedure recepteur2;
var r,s: trame;
evenement : TypeEvenement;
begin
repeat
   Attendre(evenement); /* On attend un signal */
   Origine_Couche_Physique(r}; /* Acquisition de la trame arrivee */
   Vers_Couche_Reseau(r.info); /* Transmission du paquet a la couche
reseau */
   Vers_Couche_Physique(s); /* Envoi d'acquittement */
until FinDesTemps;
end; /*recepteur2*/
```

Figure 3 : protocole send and wait.

5.3. Protocole simplex pour un canal bruité

Considérant maintenant le cas d'un canal de communication subi à des erreurs. Les trames peuvent être erronées ou perdues. Le récepteur peut détecter les trames endommagées en calculant un total de contrôle. Il n'émet pas un acquittement sauf si la trame est correctement reçue. Dans le cas échéant, l'émetteur retransmet la même trame après un temps déterminé. Le récepteur peut distinguer les trames consécutives par le numéro de séquence de trame codé sur un bit. Ce protocole est présenté dans la figure 4.

```
const MaxSeq = 1;
type TypeEvenement = (ArriveeTrame, Erreur, Timeout)
procedure emetteur3;
var PTAE: NumSequence; /*numero de la prochaine trame a envoyer */
s: trame;
tampon: paquet;
evenement: TypeEvenement;
begin
 PTAE:=0; /* initialisation */
```

```
 Origine_Couche_Reseau(tampon); /* chargement du premier paquet */
 repeat
   s.info := tampon;
   s.seq := PTAE;
   Vers_Couche_Physique(s); /* emission trame */
   StartTimer(s.seq);
   Attendre(evenement);
   if(evenement=ArriveeTrame) then
    begin
     Origine_Couche_Reseau(tampon);
     inc(PTAE); /* PTAE := (PTAE + 1) mod 2 */
    end
until FinDesTemps;
end; /* emetteur3 */
procedure recepteur3;
var TA : NumSequence; /*numero de la trame attendue */
r, s: trame;
evenement: TypeEvenement;
begin
TA:=0; /* initialisation */
repeat
  Attendre(evenement);
  if(evenement=ArriveeTrame) then
    begin
      Origine_Couche_Physique(r);
      if(r.seq = TrameAttendue) then
        begin
          Vers_Couche_Reseau(r.info);
          inc(TA); /* TA := (TA + 1) mod 2 */
        end;
      Vers_Couche_Physique(s); /* acquittement */
    end;
until FinDesTemps;
end; /* recepteur3 */
```

Figure 4 : protocole simplex pour un canal bruité

5.4. Protocole full duplex pour un canal bruité

Considérant le scénario suivant : Si par accident le récepteur met plus de temps que d'habitude à acquitter une trame, alors la temporisation de l'émetteur peut expirer avant l'arrive de l'acquittement. Mais cet acquittement n'est pas perdu et à son arrivée l'émetteur le considère comme l'acquittement de la deuxième trame I(0) qu'il a envoyée. Donc il envoie la

trame I(1). Seulement la seconde trame I(0) envoyée par l'émetteur arrive aussi au récepteur et le récepteur réagit de la même manière envoyant un acquittement. Cette fois-ci l'acquittement acquitte la trame I(1) alors que la trame I(1) peut très bien se perdre.

Le protocole dans ce cas comporte comme dans le cas précédent. Cependant il doit indiquer dans chaque acquittement le numéro de la trame qu'il attende.

Afin d'améliorer ce protocole et supposant que le récepteur puisse envoyer des informations, ce dernier peut mettre un acquittement dans une trame d'information. On appel cette technique ***superposition (Piggybacking)***.

5.5. Protocole à fenêtre d'anticipation

Le temps qui sépare l'émission d'une trame et la réception de son acquittement peut être considérable. La liaison si elle est full duplex et que le destinataire alloue un buffer pour n trames, l'émetteur peut envoyer n trames sans attendre d'acquittements. Chacune des trames doit posséder un numéro de séquence. Le destinataire renvoie un acquittement ACKi+1 pour indiquer la bonne réception de la tramei. L'acquittement porte toujours le numéro de séquence de la trame attendue par le destinataire. Cela est un signe implicite que le destinataire est prêt à recevoir une nouvelle trame (son numéro est indiqué dans l'ACK).

L'émetteur doit maintenir une liste des numéros de séquences des trames qu'il est autorisé à transmettre (définie dans la fenêtre d'émission). En plus des copies des trames qu'il a déjà envoyées et qui ne sont pas encore acquittées.

Le destinataire garde lui aussi une liste des numéros de séquences des trames qu'il peut recevoir (définie dans la fenêtre d'émission).

L'une de deux stratégies peut être appliquée si une erreur détectée dans la trame i : soit le retransmission de la trame i seulement (protocole à rejet sélectif) ou la retransmission de toutes les trames précédentes (0 .. i) (c'est-à-dire protocole à reject global).