TD5 – **Couche liaison (Datalink layer)**

{Nicolas.Ollinger, Emmanuel.Godard, Yann.Esposito}@lif.univ-mrs.fr

17 novembre 2004

☞ Les canaux de transmission imparfait entraînent des erreurs lors des échanges de données. La probabilité d'erreur sur une ligne téléphonique est comprise entre 10^{-4} et 10^{-7}. Ce TD est consacré à la compréhension des codes de détection et de correction d'erreurs. Il s'agit de méthodes misent en place au niveau de la deuxième couche OSI.
Pour l'envoie de k bits, on rajoute r bits qui permettent de tester si une erreur a eu lieu.
Dans la suite on utilisera la terminologie suivante :
- la totalité des $k + r$ bits sera appelé une *trame* ;
- la suite de k bits sera appelé le *message* ;
- la suite des r bits ajoutés au message sera appelé le *code correcteur* ou tout simplement le *code*.

1 Bit de parité

☞ Lors de l'envoie de k bits, on compte le nombre qui est égal à 1. Si ce nombre est pair on ajoute le bit 0 sinon on ajoute le bit 1 aux k bits envoyés.

1.1. Peut-on détecter si un seul bit est erroné ? deux ?

On peut détecter une erreur seulement si un nombre impair de bits on été modifié.

1.2. On considère que la probabilité d'erreur est $p = 10^{-6}$ et que $k = 7$.

(a) Donnez la probabilité qu'une trame soit erronée.

La probabilité qu'une trame soit erronée est $1 - (1 - p)^k$.

(b) Donnez la probabilité de faire un mauvais contrôle : c'est-à-dire accepter un message erroné ou ne pas accepter un message correct. En déduire une estimation du nombre moyen de bons contrôles par mauvais contrôle.

La probabilité qu'une trame possède un seule erreur sur le huitième bit : $P_1 = p(1 - p)^k$.
la probabilité qu'une trame possède exactement une erreur sur le premier bit et sur le deuxième bit mais pas sur les autres : $P_2'' = p^2(1 - p)^{k-1}$
la probabilité qu'une trame possède exactement deux erreurs : $P_2' = C_k^2 p^2 (1 - p)^{k-1}$.
la probabilité qu'une trame possède un nombre pair d'erreurs :

$$P_2 = \sum_{i=1}^{k+1/2} C_{k+1}^{2i} p^{2i} (1 - p)^{k+1-2i}.$$

Finalement la probabilité de faire un mauvais traitement est $P_1 + P_2$.
Le nombre moyen de bons contrôles par mauvais contrôle est $1/(P_1 + P_2)$

1.3. Donnez une estimation du temps moyen pour qu'une trame erronnée soit accepté sur une connexion 10Mo/s.

On envoie 10^6 trames par secondes. La probabilité de ne pas avoir d'erreur pendant x secondes est donc : $(1 - P_2)^{10^6 x}$.
Le temps moyen pour qu'un trame soit erronée est donc : $(1 - P_2)^{10^6 x} = 1/2$ ce qu'on réduit en $x = \frac{\ln(\frac{1}{2})}{10^6 \ln(1-P_2)}$

1.4. Si une erreur est détectée, que peut-on faire ?

Demander une nouvelle fois la trame. On ne peut pas la corriger.

2 Parité double

☞ Dans ce cas les trames sont considérées comme des matrices $n \times n$ (généralement $n = 8$) bits. On ajoute alors un bit de parité par colonne et par ligne.

2.1. Décrivez les avantages et les inconvénients de cette méthode par rapport à la précedente.

Avantage : s'il y a une erreur, elle est localisée par le croisement de la ligne et de la colonne donc corrigeable.
En effet supposons qu'une erreur se produise dans la matrice. Alors, l'erreur sera située le bit de parité situé sur la même ligne ainsi que le bit de parité situé sur la colonne donneront une erreur.

	X		X
	X		

Remarquons que l'on peut détecter n erreurs en rafale (qui ont eu lieu les unes après les autres).
Inconvénient : ajout de 2 bits par octets au lieu de 1.

2.2. Erreurs de contrôles

(a) Qu'est-ce que faire un mauvais contrôle dans ce cas ?

1. ne pas détecter d'erreur ;
2. faire une correction alors que ce n'est pas nécessaire.

(b) Dans quel cas ces mauvais contrôles se produisent-ils ? Quels sont les cas les plus probables ? Quelle est la probabilité pour chacun de ces cas qu'ils se produisent ?

On peut ne pas détecter d'erreur si un bit est erroné puis que ces deux bits de contrôles le sont aussi. Il faut donc au minimum 3 bits erronés pour ne pas détecter d'erreurs.
La probabilité d'avoir 1 bit erroné dans la zone de message est

$$C^1_{n\times n} p(1-p)^{n\times n}$$

sachant qu'un bit est erroné la probabilité que son bit de contrôle (ligne ou colonne) est erroné est : p.
Donc la probabilité qu'une telle erreur se produise est :

$$n^2 p^3 (1-p)^{n\times n}$$

On fait une correction qui n'est pas nécessaire seulement si deux bit contrôle l'un sur la colonne et l'autre sur la ligne sont erronés.
La probabilité d'avoir exactement deux bits erronés, l'un sur les bits de contrôles de ligne et l'autre sur les bits de contrôle de colonne est

$$C^1_n C^1_n p^2 (1-p)^{n\times n+2n-2} = n^2 p^2 (1-p)^{n\times n+2n-2}$$

3 Code de Hamming

☞ *Il s'agit d'utiliser* $\log_2 k$ *bits de parité* et d'avoir un moyen rapide de détecter la position de l'erreur.
On veut envoyer un message de m bits où $m = (2^n - 1) - n$ bits. On parle de code $x - y$ où $x = n + m$ et $y = m$. Par exemple le code de Hamming $7 - 4$ a une efficacité de $4/7 = 57\%$, $31 - 26$ efficacité de 83%. Les bits de contrôle de parité C_i sont en position 2^i. Les bits du message D_j occupent le reste de l'envoi.
Lorsque l'on veut envoyer le message $D_3 D_2 D_1 D_0$, on ajoute le code correcteur $C_2 C_1 C_0$ aux positions 2^i comme suit :

D_3	D_2	D_1	$\mathbf{C}_2$	D_0	$\mathbf{C}_1$	$\mathbf{C}_0$

Ensuite, à la réception, on recalcule les valeur des bits aux positions 2^i

D_3	D_2	D_1	$\mathbf{C'}_2$	D_0	$\mathbf{C'}_1$	$\mathbf{C'}_0$

Si $C'_2 C'_1 C'_0$ vaut 0 alors on considère qu'il n'y pas d'erreur, sinon, la valeur correspond à l'indice auquel une erreur est survenue.
Lorsque $C'_0 = 1$ alors les valeurs possibles pour $C'_2 C'_1 C'_0$ sont : 001, 011, 101 et 111 c'est-à-dire 1, 3, 5 et 7. Il s'agit des indices utilisés pour le calcul de C_0 et de C'_0.

3.1. Calculez les indices de calcul pour C_1 et C_2

C_1 : 010, 011, 110 et 111. C'est-à-dire 2, 3, 6 et 7.
C_2 : 100, 101, 110 et 111. C'est-à-dire 4, 5, 6 et 7.

Si $I_1 ... I_n$ sont les indices utilisés pour le calcul de C'_i, alors la valeur de $I_1 \oplus ... \oplus I_n$.

3.2. On souhaite envoyer le mot 1010 sachant que le mot envoyé va être de la forme

1	0	1	$\mathbf{C}_2$	0	$\mathbf{C}_1$	$\mathbf{C}_0$

Calculez les valeurs de C_0, C_1 et C_2 pour qu'à la réception les $C_2'C_1'C_0'$ soit égal à 0.

On veut que $C_2' = 0$. On sait

$$C_2' = D_3 \oplus D_2 \oplus D_1 \oplus C_2$$

Ce qui ce réécrit

$$C_2' = 1 \oplus 0 \oplus 1 \oplus C_2$$

Il faut donc $C_2 = 0$.

Faire de même pour C_1 et C_0.

3.3. Vérifiez que si une erreur à lieu, la valeur de $C_2'C_1'C_0'$ en donne la position.

4 Détection d'erreur par CRC (Cyclic Redundancy Check)

☞ On considère chaque message comme un polynôme à coëfficients dans $\{0, 1\}$. Par exemple 1101 correspond au polynôme : $x^3 + x^2 + 1$ On considère ensuite un polynôme commun à l'émetteur et au récepteur. Il s'agit du polynôme *générateur*. On va alors décaler vers la droite le message d'autant que le degrés du polynôme générateur. On va alors diviser le message décalé par ce polynôme. Il suffit alors de supprimer (concaténer au message) le reste de la division. Le récepteur fait la division et vérifie que le reste est nul.
On peut lorsque le polynôme commun est de degrés 16 détecter toutes les erreurs simples ou doubles, toutes les erreurs comportant un nombre impair de bits et tous les paquets d'erreurs de longeur inférieure ou égale à 16. Avec une très bonne probabilité, les paquets d'erreurs de longueur supérieure.
Par exemple Ethernet utilise un champs CRC à 32 bits avec le polynôme générateur :

$$X^{32} + X^{26} + X^{23}X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

4.1. Cette méthode est-elle une méthode de détection d'erreur ou de correction d'erreur ?

Cette méthode permet de détecter des erreurs mais aussi de les corriger. Le nombre de corrections possibles dépendant fortement du polynôme générateur. Ainsi, il faut choisir un bon polynôme générateur pour pouvoir utiliser efficacement ce code.

4.2. Le polynôme diviseur est : 101 c'est-à-dire x^2+1. On veut coder le mot 11010100. Quel est la trame que l'émetteur doit envoyer ?

On ne veut calculer que le reste, ainsi, il suffit de faire un XOR au fur et à mesure que la trame arrive comme dans l'exemple ci-dessous :

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

La trame à envoyer est : 11010100**11**