



## La couche physique

### Couche Liaison

1ere partie

E.Lochin

ISAE-SUPAERO

La couche physique nous permet une transmission brute de bits par un medium reliant une machine *A* à une machine *B*.

## La couche liaison (ce que dit la norme)

## La couche liaison

**7.6.2.1** La couche liaison de données fournit les moyens fonctionnels et procéduraux nécessaires d'une part à la transmission en mode sans connexion entre entités de réseau et, d'autre part, en mode connexion, à l'établissement, au maintien et à la libération des connexions de liaison de données entre entités de réseau, ainsi qu'au transfert des unités de données de service de liaison de données. Une connexion de liaison de données est construite à partir d'une ou plusieurs connexions physiques.

Source : standard UIT-T-X200

- La couche liaison est divisée en deux sous-couches
  - Sous-couche d'accès au canal (Medium Access Control), dépendante de la couche physique
  - Sous-couche indépendante de la couche physique (Logical Link Control)

## Plusieurs services

## Fonctions de la couche liaison

- Service sans connexion, sans acquittement
- Service sans connexion, avec acquittement
- Service avec connexion, avec acquittement, fiable
- Acquittement : l'émetteur sait que le message a été reçu
- Connexion : dialogue préalable pour se "connecter"
  - Téléphone vs Courrier vs Courrier+AR
- Être capable d'envoyer des messages sur le medium
- Détecter les erreurs
- Gérer les erreurs
- Gérer le flux des données

## Contenu

## Trames

- 1 Trames
- 2 Détection d'erreurs
- 3 Protocoles
- 4 En pratique

Une donnée échangée au niveau liaison s'appelle une *trame*

- Données + Commandes

Problème : comment délimiter les trames ?

## Découpage de trames

Solutions

## Codage de début et de fin

Mode octet

- Indiquer la taille de chaque trame
- Codage de début et de fin
- Reporter le problème à la couche physique

O	T	o	u	l	o	u	s	e
---	---	---	---	---	---	---	---	---

#	(header)	O	T	o	u	l	o	u	s	e	(footer)	#
---	----------	---	---	---	---	---	---	---	---	---	----------	---

On repère le début (resp. fin) de la trame par le caractère #

## Codage de début et de fin

Mode octet

## Codage de début et de fin

Mode octet

R	o	o	m	:	#	1	3	4	9
---	---	---	---	---	---	---	---	---	---

#	(header)	R	o	o	m	:	#	1	3	4	9	(footer)	#
---	----------	---	---	---	---	---	---	---	---	---	---	----------	---

R	o	o	m	:	#	1	3	4	9
---	---	---	---	---	---	---	---	---	---

#	(header)	R	o	o	m	:	!	#	1	3	4	9	(footer)	#
---	----------	---	---	---	---	---	---	---	---	---	---	---	----------	---

## Codage de début et de fin

Mode octet

## Codage de début et de fin

Mode bit

Y	E	S	!	!	I	W	o	n
---	---	---	---	---	---	---	---	---

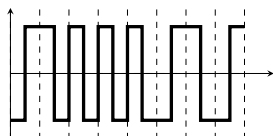
#	(header)	Y	E	S	!	!	I	W	o	n	(footer)	#
---	----------	---	---	---	---	---	---	---	---	---	----------	---

- On code le début et la fin par le fanion 01111110
- Dans tout le texte, une suite de 5 caractères 1 consécutifs est remplacée par 111110

## Codage en couche physique

## Contenu

Exemple du Codage Manchester



- 1 Trames
- 2 Détection d'erreurs
- 3 Protocoles
- 4 En pratique

- La transmission dans le medium n'est pas parfaite
- La couche physique n'offre aucune garantie

- Un code de paramètre  $(n, m)$  est un procédé qui transforme  $n$  bits en  $m > n$  bits. On note  $f(x)$  la fonction
- En général, les codes ajoutent aux  $n$  bits existants,  $m - n$  bits de contrôle. Ce n'est pas toujours le cas
- Un mot  $x$  est transmis avec  $k$  erreurs si le résultat de la transmission contient  $k$  bits différents de leur valeur initiale
- Un code détecte  $k$  erreurs si, lorsqu'un mot  $f(x)$  est transmis avec moins de  $k$  erreurs, on peut se rendre compte que le message est erroné
- Un code corrige  $k$  erreurs si, lorsqu'un mot  $f(x)$  est transmis avec moins de  $k$  erreurs, on est capable de retrouver  $f(x)$

## Premier exemple

Bit de parité

## Distance de Hamming

- On ajoute à un message de  $n$  bits un bit de parité
- Le bit vaut 0 s'il y a un nombre pair de 1 dans le message, et 1 sinon
- Detecte une erreur
- N'en corrige aucune

- La distance de Hamming entre deux mots  $x$  et  $y$  est le nombre de bits sur lesquels ils diffèrent

$$\begin{aligned} d(00010, 01010) &= \\ d(00011010, 00001011) &= \end{aligned}$$

## Distance de Hamming

## Distance de Hamming

- La distance de Hamming entre deux mots  $x$  et  $y$  est le nombre de bits sur lesquels ils diffèrent

$$\begin{aligned} d(00010, 01010) &= 1 \\ d(00011010, 00001011) &= 2 \end{aligned}$$

- La distance de Hamming entre deux mots  $x$  et  $y$  est le nombre de bits sur lesquels ils diffèrent

$$\begin{aligned} d(00010, 01010) &= 1 \\ d(00011010, 00001011) &= 2 \end{aligned}$$

- Sur un alphabet donné, on calcul le maximum vraisemblance afin de corriger de façon probabiliste

Soit l'alphabet : 00000000 00001111 11110000 11111111  
On reçoit : 00000001  
Quelle est la distance par rapport à cet alphabet ?  
Quelle est la meilleure correction ?

## Distance de Hamming

Reformulations

## Problématique

- Un code détecte  $k$  erreurs si la distance entre deux mots de code est supérieure à  $k + 1$
- Un code corrige  $k$  erreurs si la distance entre deux mots de code est supérieure à  $2k + 1$

Compromis à trouver :

- Efficacité du code : rapport  $n/m$
- Propriétés de correction du code

Codes de Hamming

- Code optimal corrigeant une seule erreur
- Les bits de contrôle sont situés aux puissances de 2
- Le bit de contrôle en position  $2^i$  est la parité de tous les bits ayant  $2^i$  dans leur décomposition en binaire

a	b	c	d
---	---	---	---

Codes de Hamming

- Code optimal corrigeant une seule erreur
- Les bits de contrôle sont situés aux puissances de 2
- Le bit de contrôle en position  $2^i$  est la parité de tous les bits ayant  $2^i$  dans leur décomposition en binaire

a	b	c	d
---	---	---	---

1	2	3	4	5	6	7
x	y	a	z	b	c	d

Codes de Hamming

- Code optimal corrigeant une seule erreur
- Les bits de contrôle sont situés aux puissances de 2
- Le bit de contrôle en position  $2^i$  est la parité de tous les bits ayant  $2^i$  dans leur décomposition en binaire

a	b	c	d
---	---	---	---

001	010	011	100	101	110	111
x	y	a	z	b	c	d

Codes de Hamming

- Code optimal corrigeant une seule erreur
- Les bits de contrôle sont situés aux puissances de 2
- Le bit de contrôle en position  $2^i$  est la parité de tous les bits ayant  $2^i$  dans leur décomposition en binaire

a	b	c	d
---	---	---	---

001	010	011	100	101	110	111
x	y	a	z	b	c	d

$$\begin{aligned}x &= a + b + d \\y &= a + c + d \\z &= b + c + d\end{aligned}$$

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

- On représente le mot  $a_i$  par le polynôme  $\sum a_i X^i$
- On calcule le résultat modulo un polynôme  $P$  (et modulo 2)
- Exemple : Calcul du CRC de 110011110 pour le polynôme  $x^5 + x^2 + 1$  (CRC-5-USB)

$$x^8 + x^7 + 0 + 0 + x^4 + x^3 + x^2 + x + 0 \mid x^5 + x^2 + 1$$

● Résultat :  $x^2 + x + 1$

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

- On représente le mot  $a_i$  par le polynôme  $\sum a_i X^i$
- On calcule le résultat modulo un polynôme  $P$  (et modulo 2)
- Exemple : Calcul du CRC de 110011110 pour le polynôme  $x^5 + x^2 + 1$  (CRC-5-USB)

$$x^8 + x^7 + 0 + 0 + x^4 + x^3 + x^2 + x + 0 \mid x^5 + x^2 + 1$$

● Résultat :  $x^2 + x + 1$

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

- On représente le mot  $a_i$  par le polynôme  $\sum a_i X^i$
- On calcule le résultat modulo un polynôme  $P$  (et modulo 2)
- Exemple : Calcul du CRC de 110011110 pour le polynôme  $x^5 + x^2 + 1$  (CRC-5-USB)

$$\begin{array}{cccccccc|c}x^8 & +x^7 & +0 & +0 & +x^4 & +x^3 & +x^2 & +x & +0 & x^5 + x^2 + 1 \\x^8 & & & & & x^5 & & & & x^3 \\& x^7 & 0 & x^5 & x^4 & 0 & x^2 & x & 0 & \\& & & & & & & & & \hline\end{array}$$

● Résultat :  $x^2 + x + 1$

Codes Polynomiaux

CRC (Cyclic Redundancy Check)

- On représente le mot  $a_i$  par le polynôme  $\sum a_i X^i$
- On calcule le résultat modulo un polynôme  $P$  (et modulo 2)
- Exemple : Calcul du CRC de 110011110 pour le polynôme  $x^5 + x^2 + 1$  (CRC-5-USB)

$$\begin{array}{cccccccc|c}x^8 & +x^7 & +0 & +0 & +x^4 & +x^3 & +x^2 & +x & +0 & x^5 + x^2 + 1 \\x^8 & & & & & x^5 & & & & x^3 + x^2 \\& x^7 & 0 & x^5 & x^4 & 0 & x^2 & x & 0 & \\& & & & x^4 & & x^2 & & & \\& & & & & x^5 & 0 & 0 & 0 & x \\& & & & & & & x & 0 & \\& & & & & & & & & \hline\end{array}$$

● Résultat :  $x^2 + x + 1$

- On représente le mot  $a_i$  par le polynôme  $\sum a_i X^i$
- On calcule le résultat modulo un polynôme  $P$  (et modulo 2)
- Exemple : Calcul du CRC de 110011110 pour le polynôme  $x^5 + x^2 + 1$  (CRC-5-USB)

$x^8$	$+x^7$	$+0$	$+0$	$+x^4$	$+x^3$	$+x^2$	$+x$	$+0$	$x^5 + x^2 + 1$
$x^8$			$x^5$		$x^3$				$x^3 + x^2 + 1$
	$x^7$	$0$	$x^5$	$x^4$	$0$	$x^2$	$x$	$0$	
	$x^7$			$x^4$		$x^2$			
			$x^5$	$0$	$0$	$0$	$x$	$0$	
			$x^5$			$x^2$		$1$	
						$x^2$	$x$	$1$	

- Faciles à calculer par un circuit électronique
- Des petits polynômes peuvent détecter beaucoup d'erreurs

- Résultat :  $x^2 + x + 1$

## Contenu

## Problématique

- 1 Trames
- 2 Détection d'erreurs
- 3 Protocoles
- 4 En pratique

Transmettre un message de  $A$  à  $B$ , en tenant compte

- Des erreurs
- Des pertes
- Du temps de traitement (couches supérieures...)

## Protocole minimal

## Protocole avec attente

- $A$  envoie toutes ses trames dès que possible  

```
while(1) physical.put(network.nextframe());
```
- $B$  reçoit toutes les trames dès que possible  

```
while(1) network.handleframe(physical.get());
```

## Problèmes

- $B$  doit être capable de traiter la trame  $i$  avant que la trame  $i + 1$  ne soit arrivée
- Ne marche pas si perte ou erreurs

- $A$  attend que  $B$  ait traité la trame avant d'envoyer la suivante

```
A: while (1)
    physical.put(network.nextframe());
    wait_acknowledgement();

B: while(1)
    network.handleframe(physical.get());
    acknowledge();
```

Pour prévenir  $A$ , le récepteur  $B$  peut utiliser le même medium :  

```
acknowledge() { physical.put(frame_ack); }
```

 ou un tout autre moyen/réseau/medium.

- Ne gère toujours pas les erreurs

## Protocole avec erreurs

Essai 1

## Protocole avec erreurs

Essai 2

- $A$  attend que  $B$  ait validé la trame avant d'envoyer la suivante

```
A: while (1)
    frame = network.frame[i];
    physical.put(frame);
    correct = physical.get();
    if (correct) i++;

B: while(1)
    frame = physical.get();
    if correct(frame)
        physical.put(1);
        network.handleframe(frame);
    else physical.put(0);
```

On suppose qu'une trame non correcte (avec une CRC invalide) n'est vue ni de  $A$  ni de  $B$  (i.e. jetée par un élément intermédiaire)

- Idem, mais  $A$  renvoie la trame si un certain délai est passé

```
A: while (1)
    frame = network.frame[i];
    physical.put(frame);
    received = physical.get() // timeout();
    if (received && !timeout) i++;

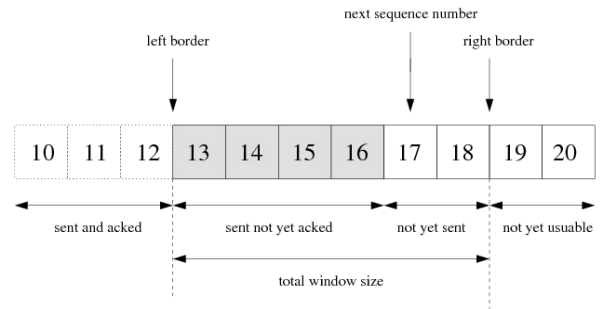
B: while(1)
    frame = physical.get();
    physical.put(1);
    network.handleframe(frame);
```

- Idem, mais  $B$  renvoie le numéro de la trame qu'il attendait.

```
A: while (1)
    frame = network.frame[i];
    physical.put(frame);
    correct = physical.get() // timeout();
    if ((correct == i) && !timeout) i++;

B: while(1)
    frame = physical.get();
    if (frame.number == i)
        physical.put(i++);
    network.handleframe(frame);
```

- $A$  envoie  $N$  frames avant d'attendre que  $B$  acquitte



## Deux derniers protocoles

## Bidirectionnel

- Go-Back-N** : taille de fenêtre  $N$ , messages numérotés de 1 à  $N + 1$
- Possibilité d'envoyer un NAK "je n'ai pas reçu le message"
- Répétition sélective** : taille de fenêtre  $N_A, N_B$ , messages numérotés de 1 à  $N_A + N_B$

- $A$  envoie des données à  $B$  et  $B$  des données à  $A$
- Le ACK est encapsulé dans les frames de données
- piggybacking*

## Contenu

- Trames
- Détection d'erreurs
- Protocoles
- En pratique

## HDLC

High-Level Data Link Control - ISO 13239

Flag	Adresse	Commande	Données	CRC	Flag
8	8	8/16	Variable	16	8

- Flag 01111110, un seul sépare deux trames
- Trois types de trames : information (i.e. données), supervision (e.g. ACK), autre
- Données : la commande contient le numéro de la trame envoyée, et celui de la trame attendue (ACK)
- Supervision : (ex.) Reject/Selective Reject : la commande contient un numéro de trame à réenvoyer

## PPP

Point to Point Protocol (RFC 1661)

## Remerciements

- Encapsulée dans HDLC
- Le champ Adresse et le champ commande ne sont pas utilisés
- Le champ données contient le protocole, et les données associées

PPP est fortement répandu et décliné (PPPoE, PPPoA, ...)

Ce cours trouve sa source dans beaucoup d'autres et notamment les slides d'Emmanuel Jeandel, de Jim Kurose et Keith Ross, ... et j'en oublie certainement.  
Merci à eux.