Cours de Réseaux

3 – La couche liaison de données

Rappel : le modèle hybride

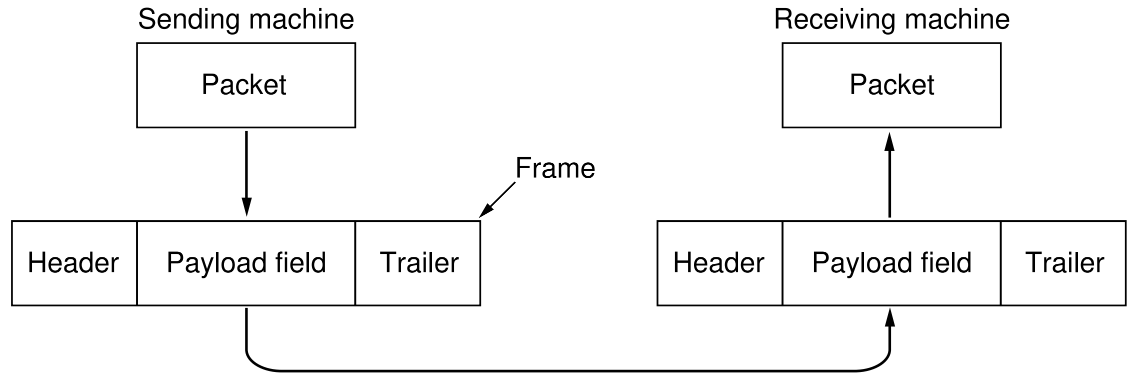
5	Couche application
4	Couche transport
3	Couche réseau
2	Couche liaison de données
1	Couche physique

Rappel : la couche liaison de données

- Gère les communications entre 2 machines adjacentes, c-à-d. directement reliées par un support physique.
- Transmission de *trames* (frames), typiquement de taille bornée.
- Traiter les erreurs de transmission (changements de bits, pertes)
- Réguler le flux des données échangées
- Dans le cas des réseaux broadcast : *Medium Access Control* (qui à le droit d'envoyer sur le réseau ?)

Les trames réseau

Envoi des informations par *trames* (frames)



Payload : *champ de données*

Découpage des trames en flux de bits (1)

Problème : déterminer le début et la fin d'une trame

(1) Faire une pause entre les trames

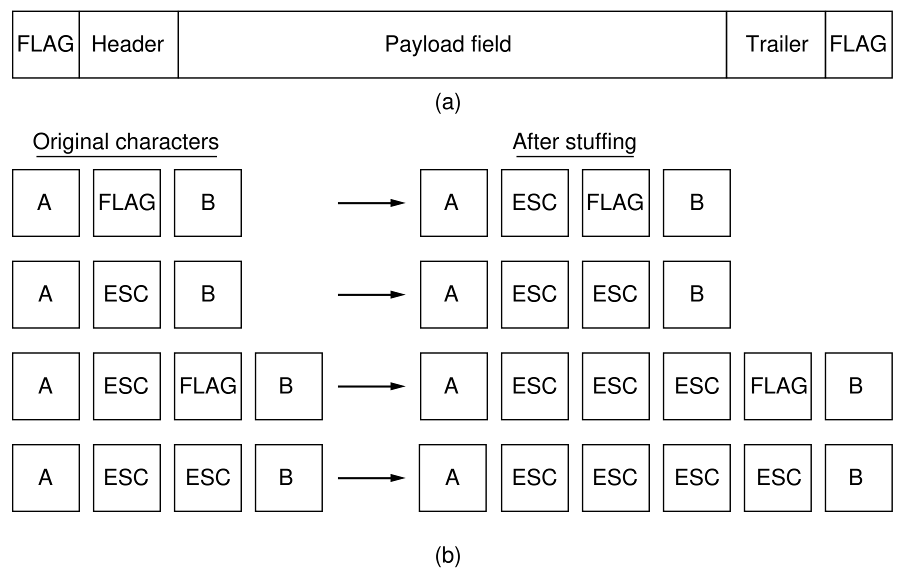
Problème : pas robuste (éléments actives dans la couche 1)

(2) Indiquer la longueur d'une trame dans l'entête

Problème : pas de moyen de re-synchroniser après une erreur de transmission

Découpage des trames en flux de bits (2)

(3) *Fanion* (flag) avec *remplissage d'octets* (byte stuffing)



Inconvénient : repose sur un format précis des caractères

Découpage des trames en flux de bits (3)

(4) Fanion avec *remplissage de bits* (bit stuffing)

Par exemple : fanion 01111110

Dans l'émission : substitution 11111 $\mapsto$ 111110

Dans la réception : substitution 111110 $\mapsto$ 11111

Découpage des trames en flux de bits (4)

(5) Utiliser un signal particulier de la couche 1 qui n'est normalement pas utilisé par la couche 1 pour envoyer des bits.

Parfois possible, mais en violation du principe d'abstraction des couches !

Erreurs de transmission

Deux stratégies :

- Codes de détection d'erreur : utilisés quand le taux d'erreurs est faible (par ex. fibre optique), et quand le récepteur peut demander une retransmission en cas d'erreur.
- Codes correcteurs d'erreurs : utilisés dans les cas des taux d'erreurs importants (par ex. WiFi) ou quand la retransmission n'est pas faisable (par ex. satellites). Aussi utilisé pour le stockage de données (CD, DVD).

Distance de Hamming

Métrique sur les n-uplets de bits :

$$d((a_1, \dots, a_n), (b_1, \dots, b_n)) = card\{i \mid a_i \neq b_i\}$$

Coder un message de m bits en un mot de *code* de n bits en utilisant une redondance de r bits, $n = m + r$.

$$c : \{0,1\}^m \rightarrow \{0,1\}^n$$

Distance de Hamming

Distance de Hamming d'un codage c : distance minimale entre deux codes.

$$\min\{d(c(a), c(b)) \mid a, b \in \{0,1\}^m\}$$

Pour détecter s erreurs il faut un code avec une distance de $s+1$.

Exemple : utiliser un bit de *parité* : distance 2, peut détecter une erreur.

Contrôle de redondance cyclique (CRC)

L'ensemble $\{0, 1\}$ a la structure d'un corps fini $GF(2)$.

Un mot de k bits $a_{k-1}, \ldots, a_0$ peut être vu comme un polynôme en $GF(2)[X]$:

$$a_{k-1}X^{k-1} + \ldots + a_1X + a_0$$

$GF(2)[X]$ est un anneau Euclidien.

Contrôle de redondance cyclique (CRC)

Choisir un polynôme $G(X)$ de dégrée $r-1$, $r \geq 2$, tel que $G_0 = 1$ (et $G_{r-1} = 1$).

Codage du mot a de longueur m (vu comme le polynôme $A(X)$ de dégrée $m-1$), $n > r$:

$$A(X) \cdot X^r + \big((A(X) \cdot X^r) \bmod G(X)\big)$$

où $x \bmod y$: reste de la division de x par y.

Contrôle de redondance cyclique (CRC)

Sur réception d'un message t de longueur $n = m + r$ (vu comme un polynôme $T(X)$ de dégrée $n - 1$) :

Vérifier que $G(X)$ divise $T(X)$.

Détecte toutes les erreurs $E(X)$ sauf celles qui sont divisibles par $G(X)$.

Contrôle de redondance cyclique (CRC)

Erreur d'une bit : $E(X) = X^i$: détectée car $G(X)$ ne divise pas $E(X)$.

Erreur de deux bits : $E(X) = X^i + X^j = X^j(X^{i-j} + 1)$: $G(X)$ n'est pas divisible par X, donc il suffit d'assurer que $G(X)$ ne divise pas $X^i + 1$ pour certaines valeurs de i.

Par exemple : $X^{15} + X^{14} + 1$ ne divise pas $X^i + 1$ pour $i < 32768$.

Contrôle de redondance cyclique (CRC)

Erreur de parité : On choisit $G(X)$ tel que $X+1$ divise $G(X)$. $E(X)$ a un nombre impair de monômes. Or, un tel polynôme ne peut pas être divisible par $X+1$ (Preuve : $E(1)=1$ si $E(X)$ a un nombre impair de monômes, contradiction avec le fait que la valeur de $X+1$ à l'endroit $X=1$ est 0).

Erreur en rafale (burst erreur) de longueur s : Si $s<r$ alors $G(X)$ ne divise pas $E(X)$.

Codes correcteurs d'erreurs

Code naïve : répéter chaque bit trois fois :

$$x_1x_2\ldots \mapsto x_1x_1x_1x_2x_2x_2\ldots$$

Réception : sur 3 bits, prendre la valeur majoritaire.

Inconvénients :
– Très cher ($r = 2 * m$).
– Pas robuste contre des erreurs en rafale.

Pour corriger s erreurs il faut un code de distance $\geq 2s + 1$.

Codes correcteurs d'erreurs

Valeur minimale de r pour corriger une seule erreur, étant donnée m :

$$\begin{aligned}(n+1) * 2^m &\leq 2^n \\ m + r + 1 &\leq 2^r\end{aligned}$$

Le code de Hamming

Soit $m + r + 1 \leq 2^r$. On représente $a_1, \ldots, a_m$ sur $n = m + r$ bits comme $b_1, \ldots, b_n$.

Soit $bin(i, j)$ le i-ième bit de la représentation binaire de j.

– Les bits à des positions 2^i sont des bits de parité
– Les autres sont des bits de message, remplis dans l'ordre avec $a_1, a_2, a_3, \ldots$.
– Pour tout i : mettre p_i tel que la parité des bits qui se trouvent à des positions j telles que $bin(i, j) = 1$, est paire.

Le code de Hamming

Décodage :

- Initialiser *compteur* $:= 0$
- Pour chaque bit sur position 2^i vérifier sa parité
  Si parité n'est pas bonne : $compteur := compteur + 2^i$
- À la fin : Si $compteur > 0$ alors erreur à la position *compteur*

Le code de Hamming

Exemple : Message 0110101. On a $m = 7$, donc $r = 4$

Bit de parité	Parité des bits au positions suivantes est paire
p_1	1,3,5,7,9,11
p_2	2,3,6,7,10,11
p_3	4,5,6,7
p_4	8,9,10,11

Codage : $\underline{10}0\underline{0}110\underline{0}101$

Codes linéaires

Codage linéaire : donné par G une matrice $m \times n$.

Fonction de codage : $x \mapsto x \cdot G$ (x vecteur de ligne).

Codes linéaires

Exemple Hamming avec $m = 7, n = 11$:

$$G = \begin{pmatrix} \color{red}{1} & \color{red}{1} & 1 & \color{red}{0} & 0 & 0 & 0 & \color{red}{0} & 0 & 0 & 0 \\ \color{red}{1} & \color{red}{0} & 0 & \color{red}{1} & 1 & 0 & 0 & \color{red}{0} & 0 & 0 & 0 \\ \color{red}{0} & \color{red}{1} & 0 & \color{red}{1} & 0 & 1 & 0 & \color{red}{0} & 0 & 0 & 0 \\ \color{red}{1} & \color{red}{1} & 0 & \color{red}{1} & 0 & 0 & 1 & \color{red}{0} & 0 & 0 & 0 \\ \color{red}{1} & \color{red}{0} & 0 & \color{red}{0} & 0 & 0 & 0 & \color{red}{1} & 1 & 0 & 0 \\ \color{red}{0} & \color{red}{1} & 0 & \color{red}{0} & 0 & 0 & 0 & \color{red}{1} & 0 & 1 & 0 \\ \color{red}{1} & \color{red}{1} & 0 & \color{red}{0} & 0 & 0 & 0 & \color{red}{1} & 0 & 0 & 1 \end{pmatrix}$$

Les colonnes en rouge donnent le calcul des bits de parité.

Codages systématiques

En peut réarranger le codage et mettre les bits de parité à la fin du code :

$$G' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Codages systématiques

La matrice d'un codage systématique est de la forme

$$G' = (I_m, P)$$

où P est une matrice $m \times r$, qui détermine le calcul des r bits de *contrôle* (ou parité).

Étant donné un message $y \in \{0,1\}^n$ on calcule le *syndrome* :

$$y \cdot P$$

Exemple Hamming :
syndrome = position du bit erroné en notation binaire.

Codages linéaires

L'art de la conception de code linéaires consiste en

trouver des matrices P telles qu'on peut facilement calculer l'erreur à partir d'un syndrome.

Quelques noms des codes linéaires célèbres :
- code de Hamming
- code de Reed-Muller
- code(s) de Golay (utilisé dans la transmission d'images des sondes spatiales Voyager)

Les codes cycliques

- Basés sur les polynômes à coefficients venant d'un corps fini (comme CRC, mais pour corriger des erreurs il faut aussi résoudre des équations entre polynômes)
- Plus robuste contre des erreurs en rafale que les codes linéaires
- Exemple : Code de Reed-Solomon
  (utilisé dans les archives PAR2, dans une forme simplifiée dans les formats de stockage sur CD/DVD)

Un protocole naïf de liaison de données

Le cas le plus simple :

– Canal parfait (pas d'erreurs de transmission, pas de perte de trames)
– Le récepteur peut traiter les trames reçues à une vitesse non-bornée

L'algorithme est très simple :

– L'émetteur, dans une boucle infinie, demande des paquets à la couche réseau, les encapsule dans une trame, et passe la trame à la couche physique.
– Le récepteur reçoit les trames, en extrait les paquets, et passe les paquets à sa couche réseau.

Le simulateur de protocoles

Interface avec la couche réseau :

- `from_network_layer : unit -> string` obtenir un nouveau paquet (type `string`) de la couche réseaux
- `to_network_layer : string -> float -> unit` délivrer un paquet à la couche réseau, qui le traite avec un certain délai (0 dans le premier cas)

Le simulateur de protocoles

Interface avec la couche physique :

- `(create_channel frame speed delay loss damage)` renvoie
  - une procédure pour envoyer des trames
  - un descripteur de signal (condition) pour l'arrivée d'une trame
  - la référence d'une variable qui contient la trame arrivée
  - un verrou (mutex) associé à cette variable

Démo : protocole 1

```
(* Protocol 1: assumes a perfect channel (no damage, no loss).  *)
(* The network layer on the receiving side can process packages *)
(* at unlimited speed.                                          *)

#use "simulator.ml"

type frame = { payload : string };;
let dummyframe = { payload = "" };;

let (send,receive,output_trigger,output_lock) =
  create_channel dummyframe 1.0 5.0 0 0
;;
```

```
let create_sender frame_send =
  let frame_of_paquet p = { payload = p }
  in
  while true do
    frame_send (frame_of_paquet (from_network_layer ()))
  done
;;

let create_receiver (receive,trigger,lock) =
  let paquet_of_frame f = f.payload
  in
  Mutex.lock lock;
  while true do
    Condition.wait trigger lock;
    match receive () with
```

```
Some f -> to_network_layer (paquet_of_frame f) 0.0
      | None -> failwith "package cannot be dammaged"
  done
;;

begin
  ignore (Thread.create create_receiver (receive,output_trigger,output_lock));
  ignore (Thread.create create_sender send)
end
```

Un protocole avec contrôle de flux (1)

– Canal parfait
– Le récepteur ne peut pas traiter les trames à une vitesse arbitraire

Solution simple :
– Le récepteur émet un acquittement quand il est prêt à accepter la trame suivante
– L'émetteur attend l'acquittement avant d'envoyer la trame suivante

Implémentation : voir le TP 2.

Un protocole avec contrôle de flux (2)

Ce protocole nécessite un canal half-duplex (ou deux canaux simplex) :

- *simplex* : transmission dans une seule direction
- *half-duplex* : transmission dans les deux sens mais de façon exclusive
- *duplex* : transmission simultanée dans les deux sens

Inconvénients de l'algorithme :

- Gaspillage de la capacité du canal
- Ne marche pas en présence de pertes

Un protocole pour un canal bruité (1)

Effet d'une erreur de transmission ou perte dans l'algo précédent sur la voie émetteur → récepteur :

La réception de la trame ne sera pas acquittée.

Solution : ajouter un timer côté émetteur
Si le timer déclenche alors l'émetteur renvoie de nouveau la trame.

Un protocole pour un canal bruité (2)

Effet d'une erreur de transmission ou perte dans l'algo précédent sur la voie récepteur → émetteur :

Le récepteur a reçu la trame et délivré le paquet à sa couche réseau.
Le timer de l'émetteur déclenche, donc la trame sera envoyé de nouveau.
La couche réseau côté récepteur reçoit un paquet en double.

Un protocole pour un canal bruité (3)

Solution :

Ajouter un compteur dans les trames
L'émetteur incrément le compteur quand il reçoit un paquet de sa couche réseau
Le récepteur jette les trames avec des numéros déjà vus.

Un compteur modulo 2 est suffisant

Protocoles avec fenêtre d'anticipation (1)

(Sliding window protocols) Permet une meilleure utilisation de la capacité du canal physique.

L'émetteur et le récepteur utilisent chacun une *fenêtre*.

Une fenêtre de taille n contient toujours des trames avec numéros $i, i+1, \ldots, i+n-1$ (éventuellement avec des trous).

Protocoles avec fenêtre d'anticipation (2)

Côté émetteur :

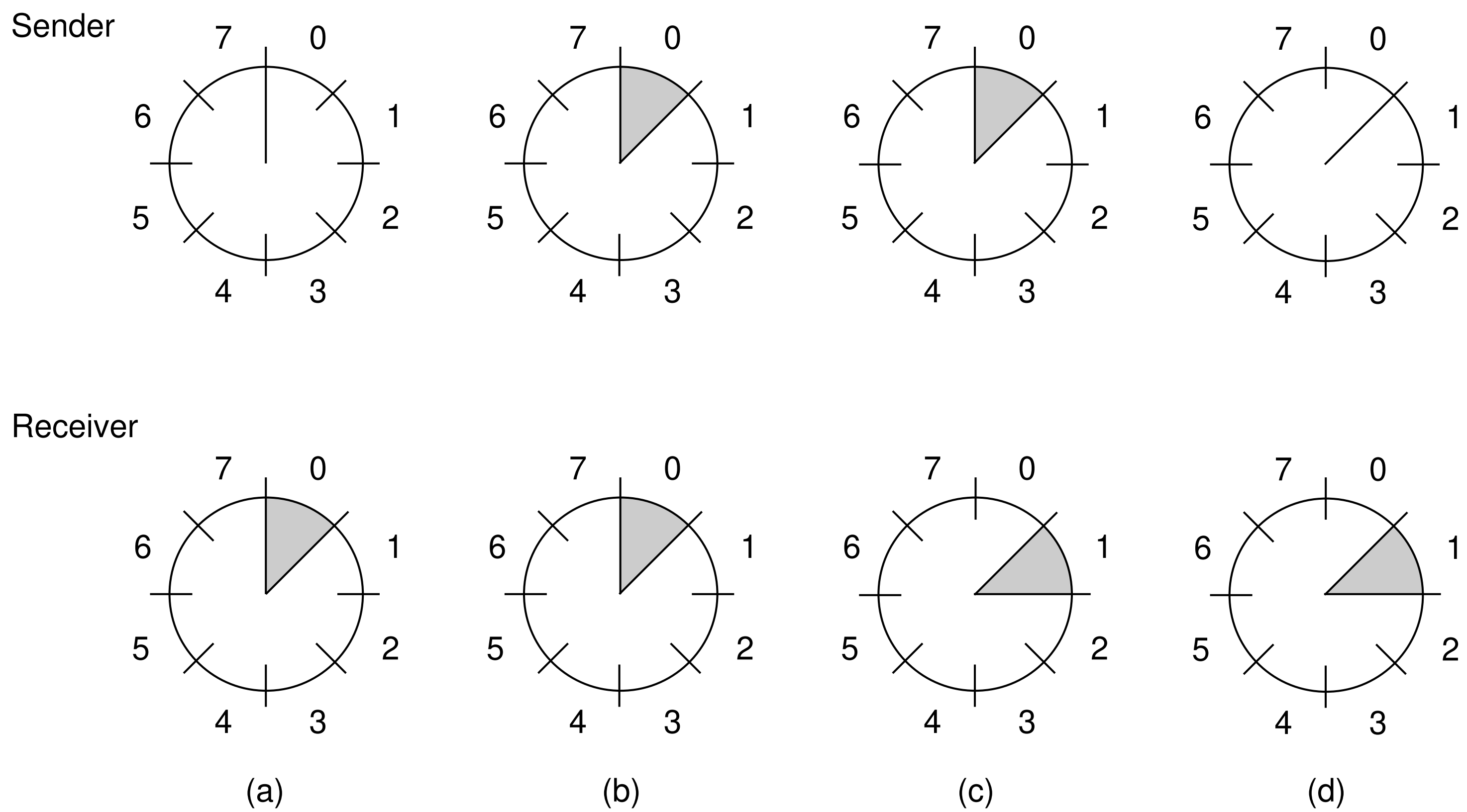
– La fenêtre de l'émetteur couvre les numéros des trames envoyées mais pas encore acquittées. Taille variable (mais typiquement bornée).
– Quand il prend un nouveau paquet de la couche réseau il place la trame en haut de la fenêtre (et avance donc la borne supérieure)
– Quand la trame la plus basse de la fenêtre est acquittée : avancer la borne inférieure de la fenêtre

Protocoles avec fenêtre d'anticipation (3)

Côté récepteur :

- la fenêtre du récepteur couvre les numéros des trames qu'il est prêt à accepter. Taille fixe.
- Quand il reçoit une trame avec le numéro en bas de sa fenêtre il transmet le paquet à sa couche réseau (ainsi que les trames successeurs qu'il a déjà dans sa fenêtre), et avance les *deux* bornes de sa fenêtre.

Protocoles avec fenêtre d'anticipation (4)



(a) Initial (b) après envoi (c) après réception (d) acquittement reçu

Protocoles avec fenêtre d'anticipation (6)

Acquittements :

– Portent un numéro i

– Sémantique : toutes les trames jusqu'au numéro i ont été reçues

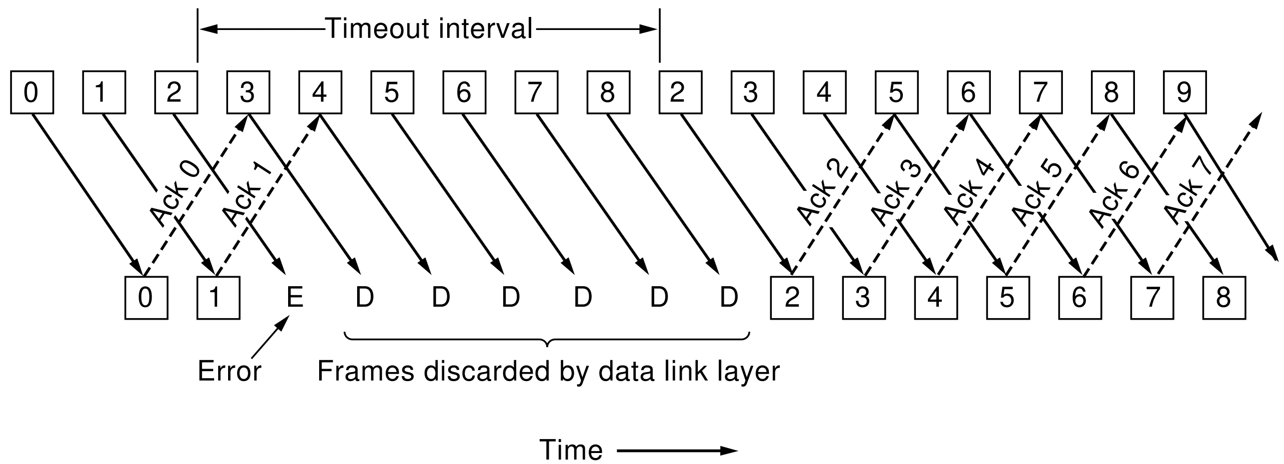
– Optimisation : acquittement négatives en cas d'erreur

Deux stratégies en cas d'erreur :

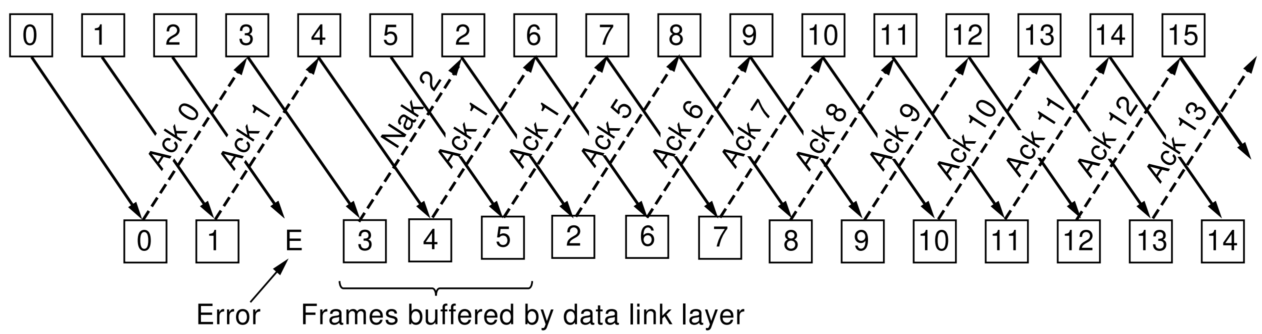
– *rejet global* : en

– *rejet sélectif*

Protocoles avec fenêtre d'anticipation (6)



(a)



(b)

Protocoles avec fenêtres d'anticipation (6)

Soient

– fenêtre émettrice : $i, \ldots, i+n$

– fenêtre réceptrice : $j, \ldots, j+m$

Invariants :

– $i \leq j$

– $j \leq i+n+1$

Si on compte les trames modulo k :

Il faut $k \leq n+m$ (sinon ambiguïté)

Résumé : structure d'une trame

Bits	8	8	8	⩾ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

Dans un vrai protocole il y a aussi

– *Superposition* (piggybacking) des acquittements et des messages
– Message de contrôle de la couche liaison de données (LCP : Link Control Protocol) : activer et fermer une ligne, tester, négocier les paramètres de la connexion
– Message de contrôle pour initialiser la couche réseau (NCP : Network Control Protocol)