

# La Couche Liaison

Modèle OSI : couche 2

# La Couche Liaison

**Objectif :** assurer une communication fiable et efficace entre deux machines adjacentes, ie les données échangées par la couche réseau doivent être :

- dans l'ordre, sans erreur, sans perte , et sans duplication.

**Principale fonctionnalité :**

- Transmission 'séquentielle' des données sous formes de trames confiées à la couche physique

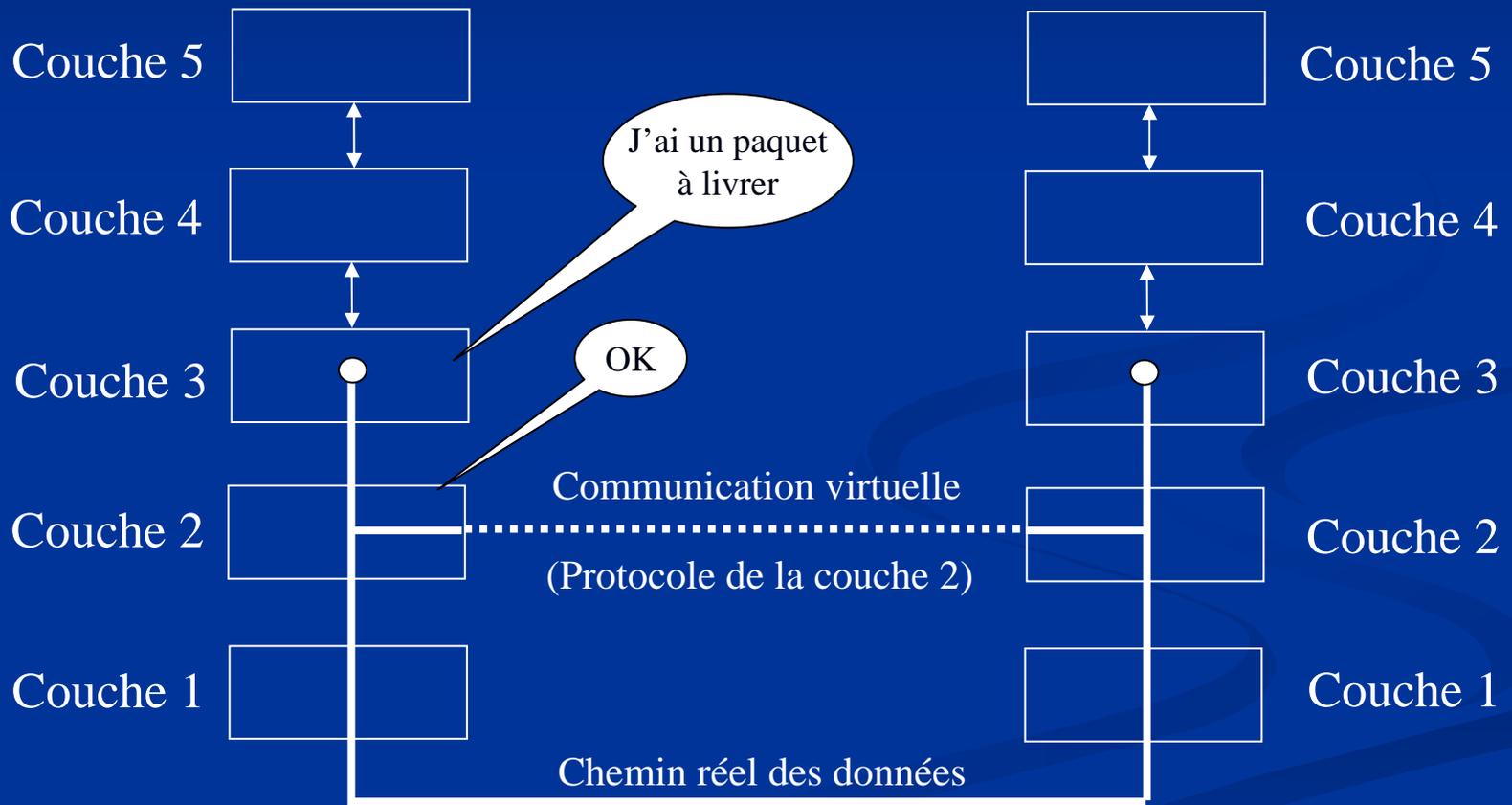
■ **Chemin virtuel :**

couche 2  $\leftrightarrow$  couche 2 (trames)

■ **Chemin réel :**

couche 1  $\leftrightarrow$  couche 1 (suite de bits sur canal de transmission: voir cours précédent...)

# Communication virtuelle



# Le contrôle d'erreurs

## ■ Considérations :

- Le canal de transmission délivre les bits dans l'ordre d'émission, *mais* certains peuvent *changer de valeur*, ou *disparaître*, ou *apparaître*.
- Une trame doit être délivrée 1 et 1 seule fois à la couche réseau destination.

→ calcul d'une somme de contrôle d'erreurs (CRC), acquittements, temporisateurs, numérotation des trames.

# Autres rôles de la couche liaison

- **Le contrôle de flux** : l'émetteur ne doit envoyer des trames que si le récepteur est en mesure de les traiter.
- **La gestion de la liaison** :
  - établissement et libération de la liaison,
  - supervision du fonctionnement selon le mode de synchronisation, de transmission, et le type de liaison,
  - définition de la « syntaxe » des trames et du protocole de liaison.

# L'utilité de la couche liaison

**Exercice** : Un message provenant des couches hautes est découpé en 10 trames, chacune d'entre elles ayant 80% de chances d'arriver intacte. Si la couche liaison n'effectue aucun contrôle d'erreur, combien de fois, en moyenne, faudra-t-il réémettre le message pour qu'il arrive entier en bon état ?

Chances que le message soit correct :  $\left(\frac{80}{100}\right)^{10} \approx \frac{1}{10}$

→ en moyenne, il faudra envoyer 10 fois le message.

# Les protocoles ARQ

**ARQ (Automatic Repeat reQuest)** : l'émetteur attend des acquittements positifs ou négatifs ; le récepteur détecte les erreurs, et selon le cas, ignore la trame ou demande sa retransmission.

Deux types de protocoles ARQ :

- protocoles « **envoyer et attendre** » (send and wait),
- protocoles « **continus** » (continuous ou pipelined ARQ) ou « **à fenêtre d'anticipation** ».

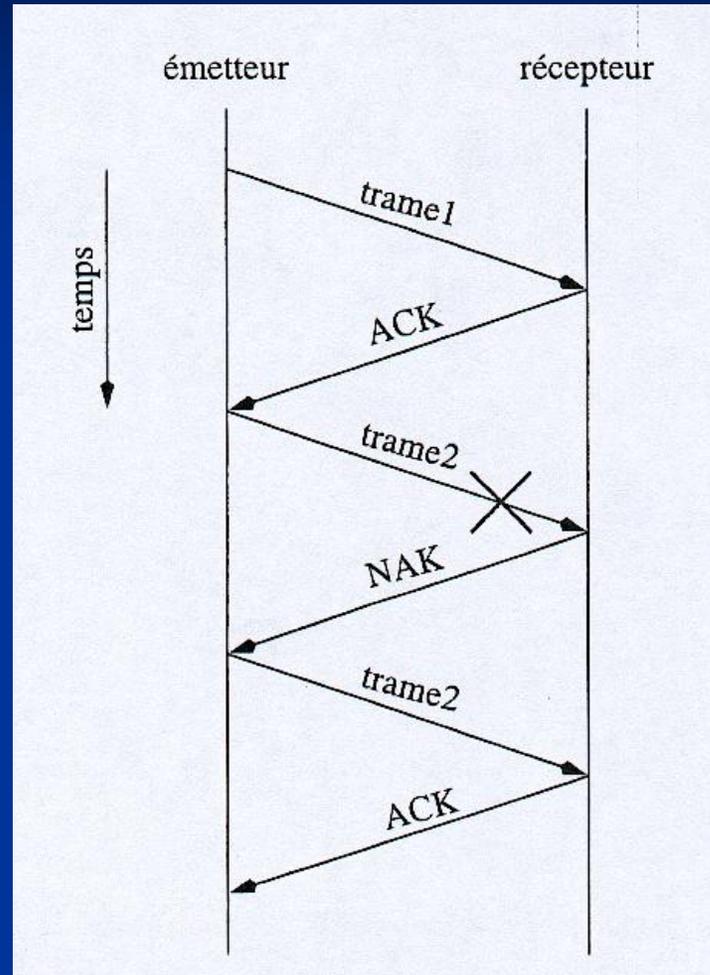
# Protocoles « envoyer et attendre »

## ■ Méthode :

- Le récepteur informe l'émetteur des situations d'erreur  
→ acquittements.
- L'émetteur envoie et attend l'information du récepteur...

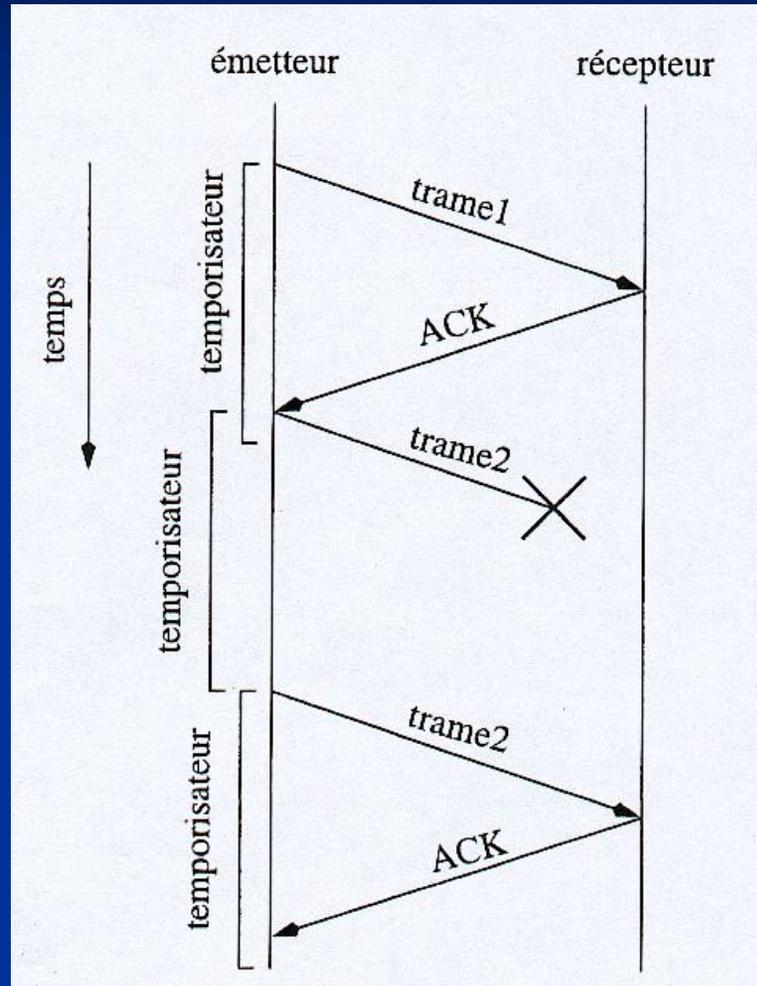
## ■ Conséquence : empêche l'émetteur d'envoyer des données plus rapidement que le récepteur ne peut les traiter.

# Protocoles ARQ (1)



Trame erronée : acquiescement positif ou négatif

# Protocoles ARQ (2)

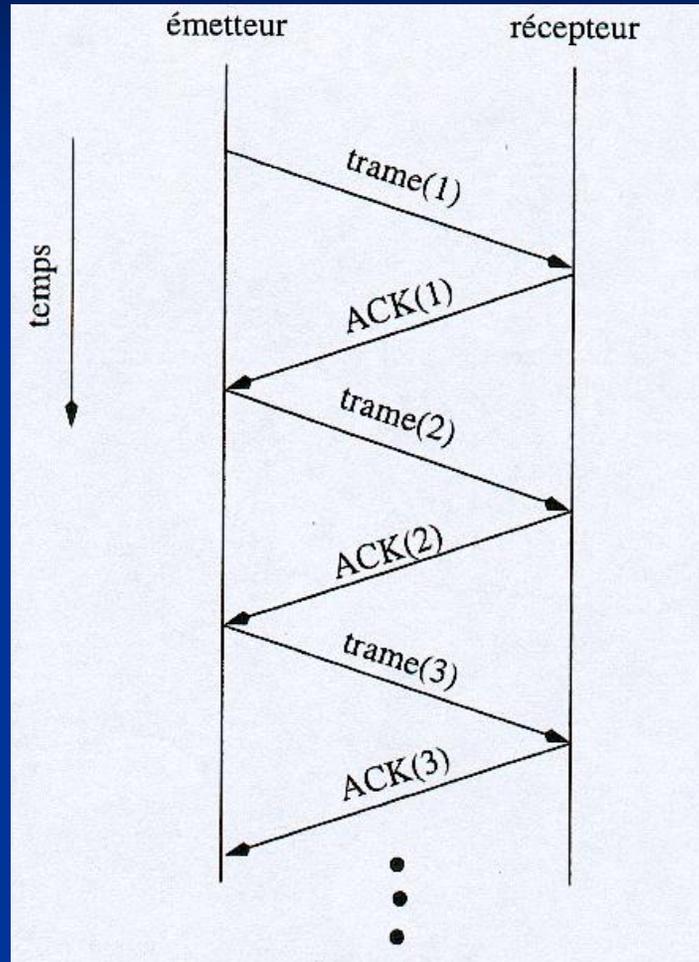


Trame perdue : temporisateur





# Protocoles ARQ envoyer et attendre



Ces protocoles sont unidirectionnels, et ne permettent qu'une pauvre utilisation de la capacité du canal.

# Protocoles « à fenêtre d'anticipation »

## Améliorations :

- Données (et acquittements) dans les 2 sens (mode bidirectionnel).
- Envoi d'un certain nombre de trames sans attendre d'acquiescement (pipelining)
- Acquiescements ajoutés à des trames de données envoyées dans l'autre sens (piggybacking).

→ + d'efficacité, + de complexité de gestion aussi

→ besoin de tampons pour trames non encore acquiescées (et susceptibles d'être réémises).

# Fenêtre d'anticipation

**Trames** : ont un numéro de séquence codé sur  $n$  bits ( $0 \rightarrow 2^n - 1$ ).

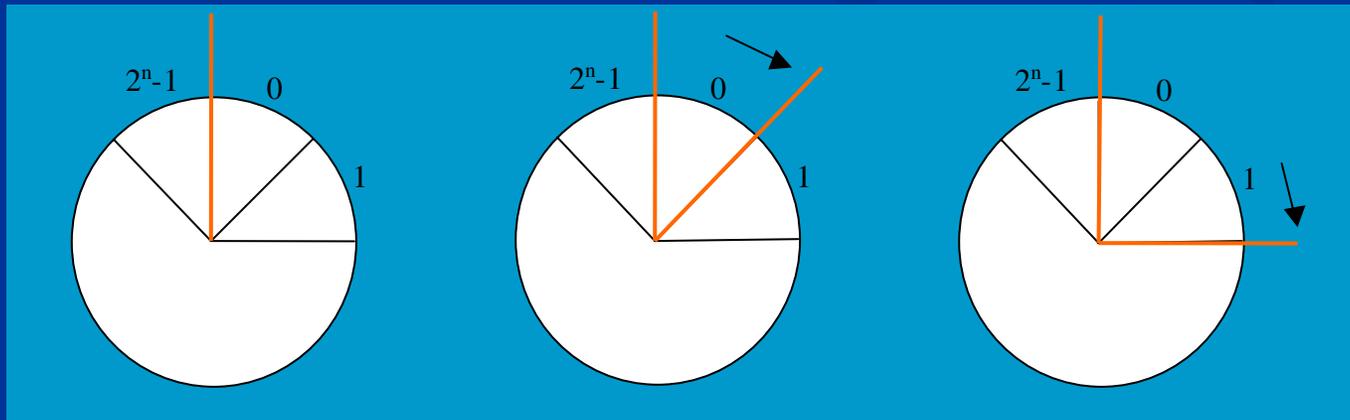
Remarque : si  $n=1 \Rightarrow$  « envoyer et attendre » robuste.

- **Fenêtre d'émission** (côté émetteur) :  
liste des numéros de séquence des trames autorisées à être *émises* .
- **Fenêtre de réception** (côté récepteur) :  
liste des numéros de séquence des trames autorisées à être *reçues* .

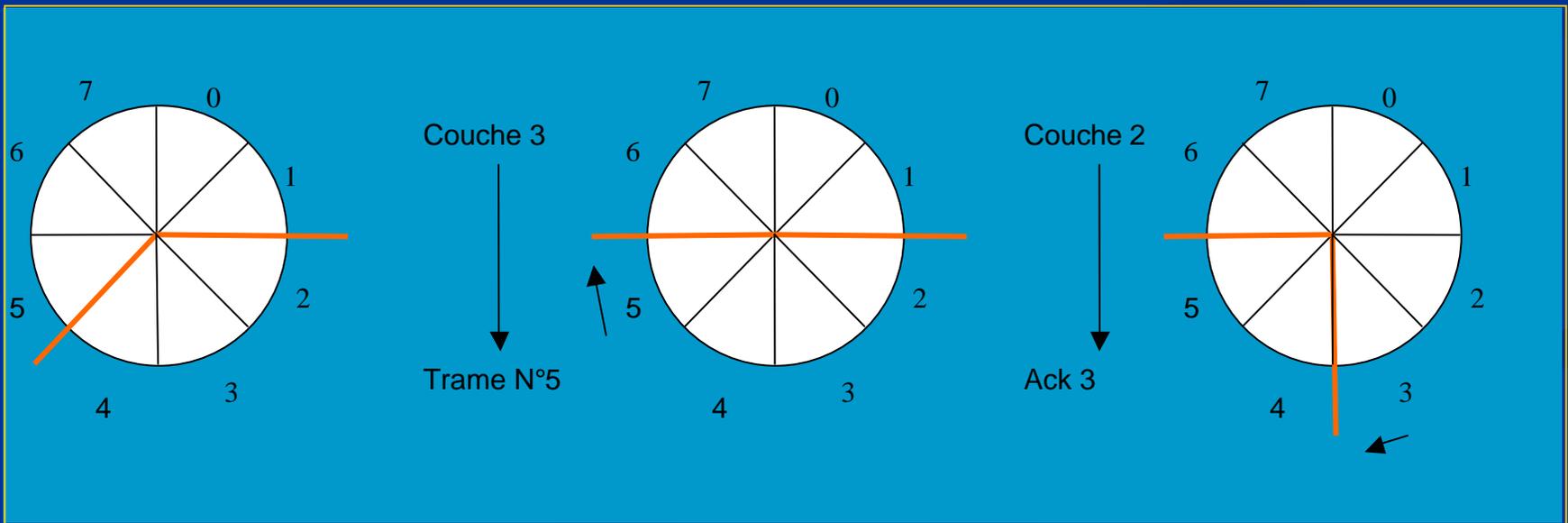
# Fenêtre d'émission

**Taille (maximale)** = nombre de trames autorisées à être émises sans attendre acquittement.

**Contenu** = numéros de séquence des trames envoyées mais non encore acquittées.



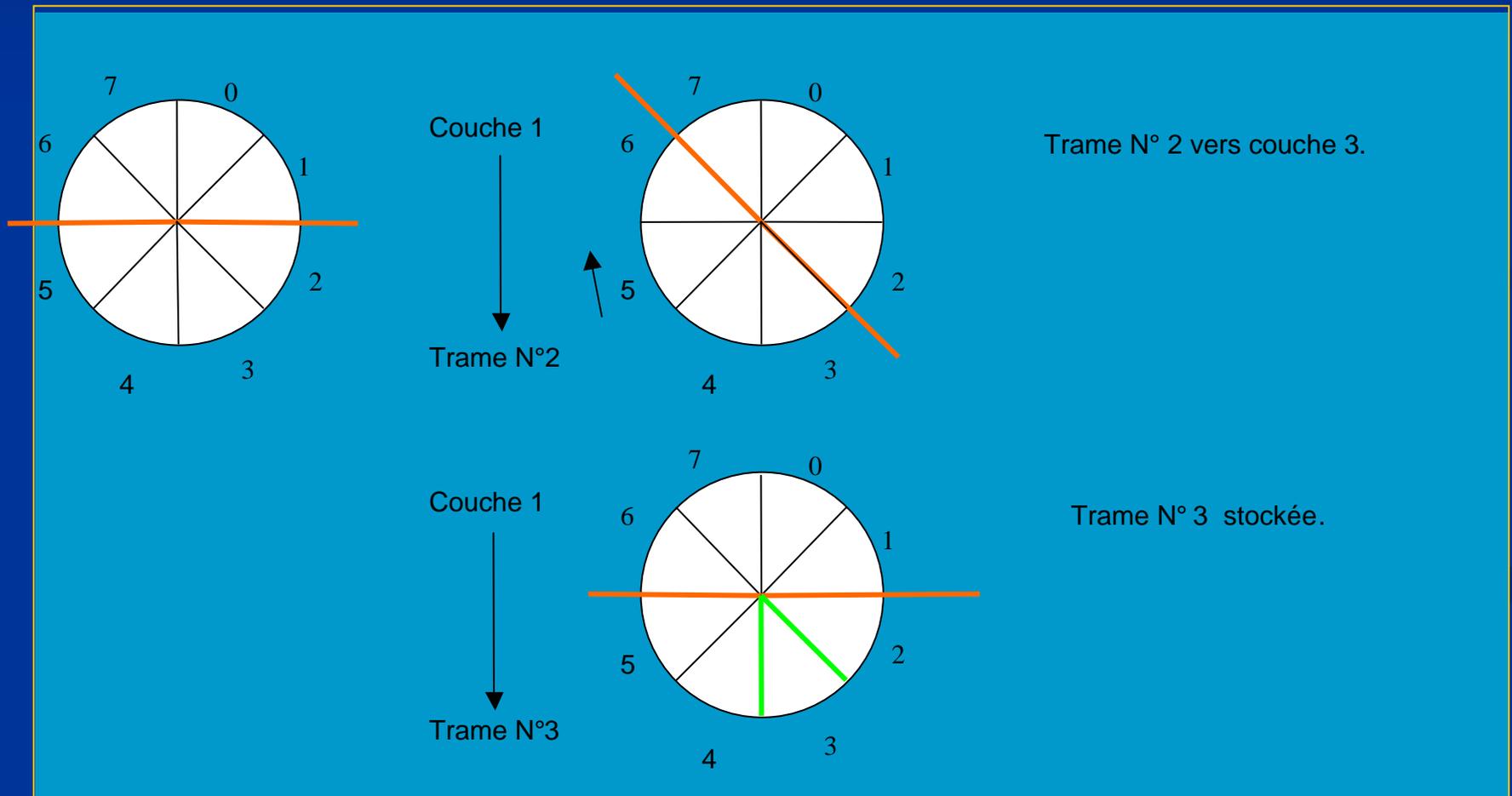
- L'émetteur stocke les trames non acquittées dans des zones tampons (au plus  $m$  trames, si  $m$  est la taille de la fenêtre).
- Si la fenêtre atteint son maximum, on n'envoie plus rien jusqu'à une libération, ie un acquittement.



# Fenêtre de réception

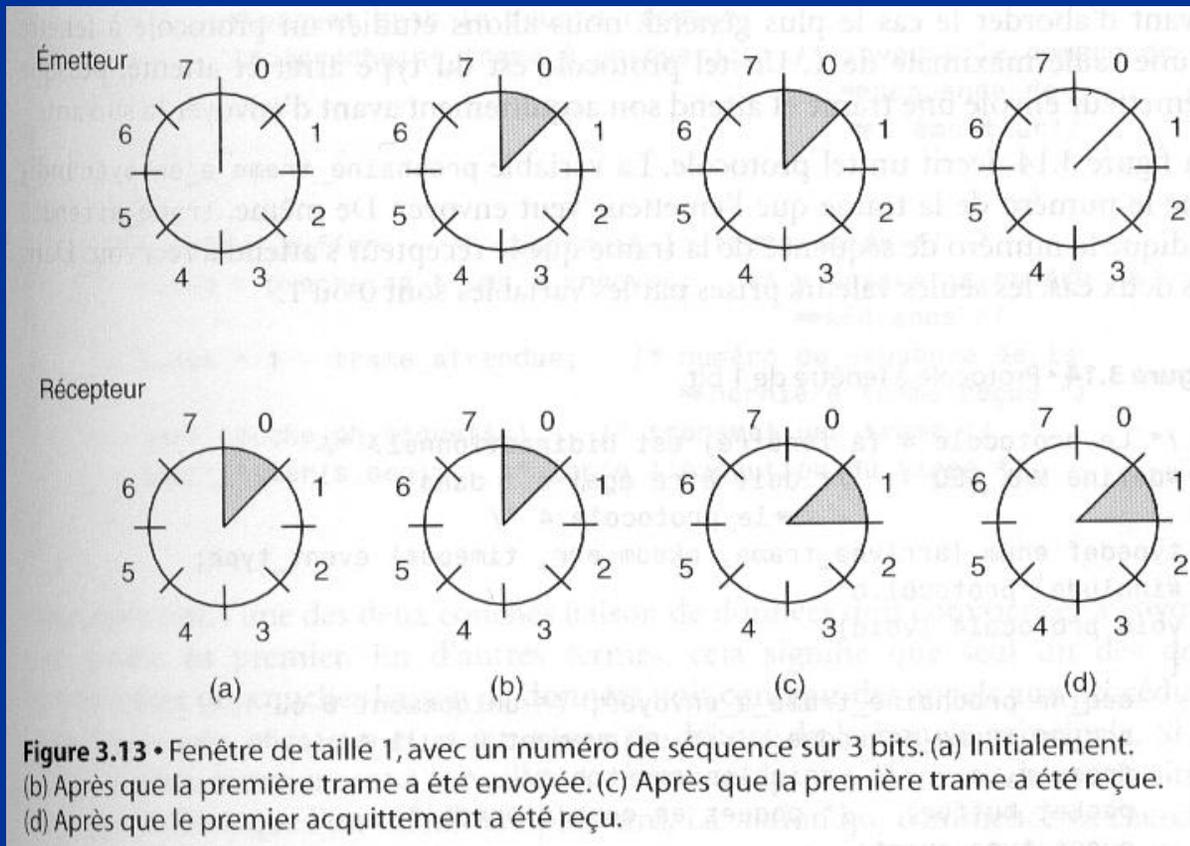
**Taille (fixe)** = nombre de trames autorisées à être reçues.

**Contenu** = numéros de séquence de ces trames attendues.



# Exemples de protocoles (1)

- Protocole à fenêtres d'émission et de réception de largeur 1  
→ revient à « envoyer et attendre » robuste.



# Exemples de protocoles (2)

- Protocole à fenêtre d'émission de largeur  $m$  et à fenêtre de réception de largeur 1

→ rejet global

- L'émetteur peut envoyer plusieurs trames sans acquittement (jusqu'à  $m$ ).
- Le récepteur rejette toutes les trames qui suivent une trame erronée.

→ retransmission de toutes les trames qui suivent celle erronée.



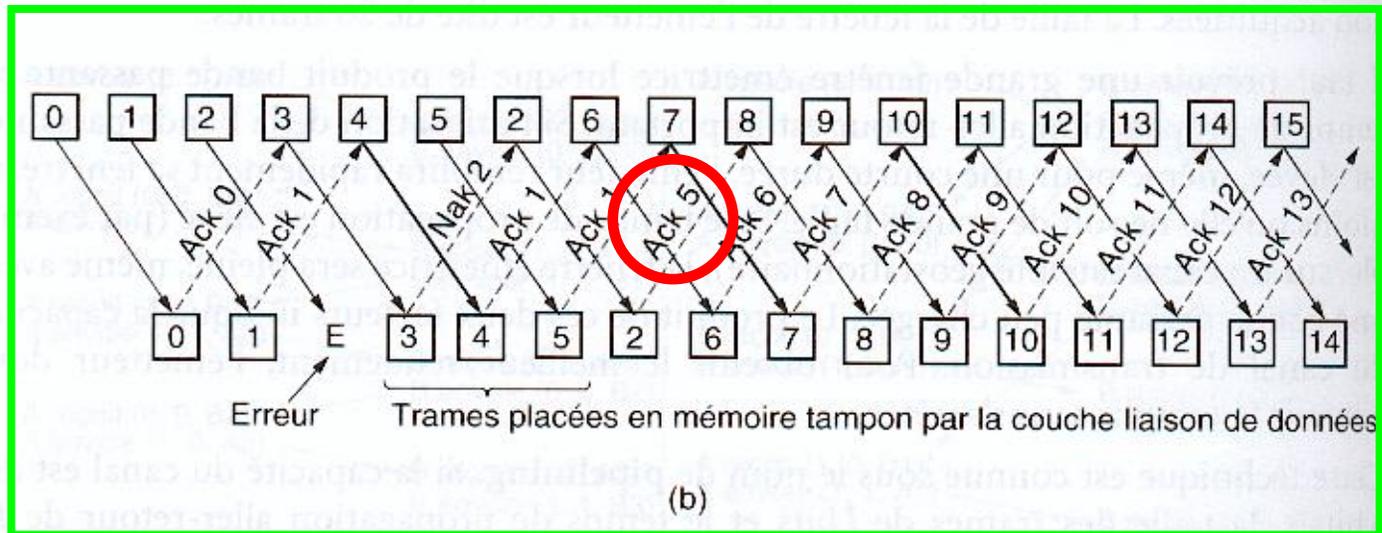
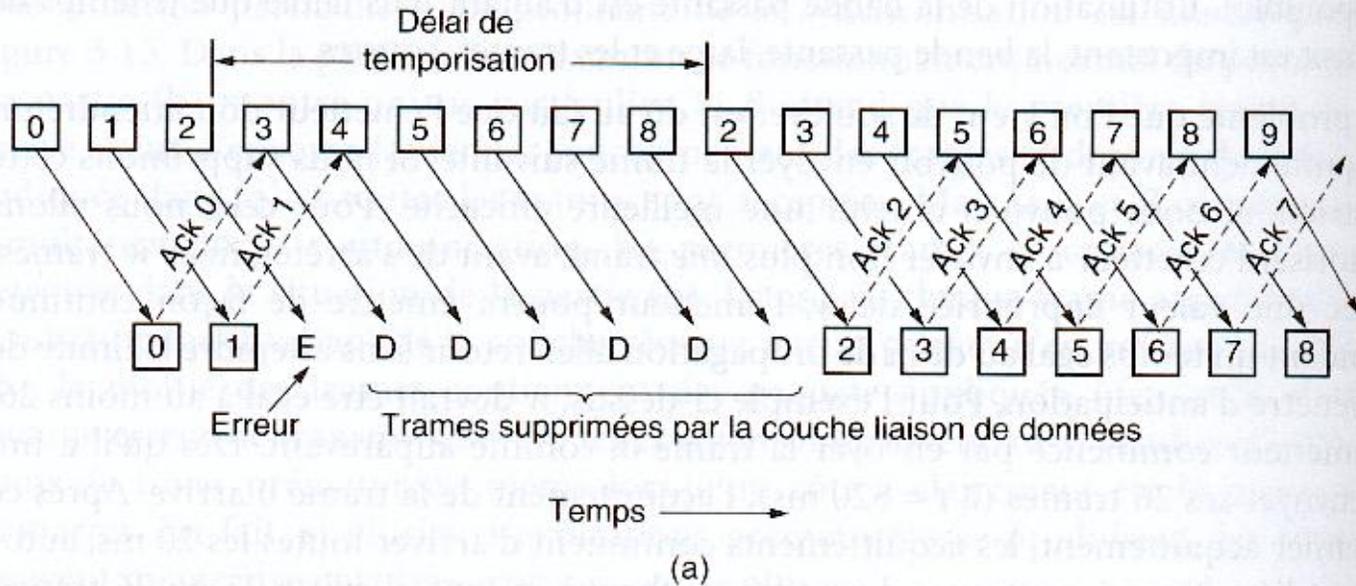
# Exemples de protocoles (3)

- Protocole à fenêtre d'émission de largeur  $m$  et à fenêtre de réception de largeur  $k$

→ rejet sélectif

- Le récepteur stocke les suivantes.
- Quand le récepteur reçoit la trame qui manquait, il envoie un acquittement du + grand nombre possible.

→ moins de retransmission de trames.



**Figure 3.16** • Pipelining et récupération d'erreur. Effet d'une erreur lorsque (a) la fenêtre du destinataire a une taille de 1 et (b) lorsqu'elle est de taille importante.

# Utilisation des fenêtres d'anticipation

L'utilisation des fenêtres d'anticipation pour les protocoles avec une transmission continue doit respecter les conditions suivantes :

- un temporisateur individuel pour chaque trame non acquittée,
- taille des fenêtres et numéros de séquence :  
on a  $t_e \geq t_r$  et
  - si  $t_r = 1$ , alors  $t_e \leq 2^n - 1$ ,
  - si  $t_r > 1$  (rejet sélectif), alors  $2 * t_e \leq 2^n$ .

# Deux familles de protocoles synchrones

## ■ Protocoles basés sur le caractère :

trame = suite de caractères.

Exemple :

- BSC (Binary Synchronous Communications) d'IBM.

→ exploitation half-duplex, utilisation d'un code (ex. ASCII).

## ■ Protocoles basés sur l'élément binaire :

trame = suite de bits.

Exemples :

- SDLC (Synchronous Data Link Control) d'IBM,
- HDLC (High-level Data Link Control) de l'ISO.

# Le protocole HDLC

- Exploitation full-duplex de la liaison.
- Basé sur l'élément binaire :
  - pas d'interprétation du contenu,
  - transparence / aux codes éventuellement utilisés.
- Protocole synchrone :
  - synchro-bit : horloge,
  - synchro-trame : délimiteur ou fanion 01111110.
  - bit de transparence (un 0 après cinq 1).
- Protection contre les erreurs de transmission pour chaque trame.
- Une trame contient données et/ou infos de service (ex. ACK).

# Types de liaison

- **Liaison non-équilibrée** : point-à-point ou multipoint.
  - La primaire commande, la secondaire répond .
  - La primaire gère la liaison (activation/désactivation).
  
- **Liaison équilibrée** : point-à-point.
  - Stations mixtes primaire/secondaire, commandes et réponses.

# Modes de fonctionnement des stations

définissent 3 classes de protocoles HDLC.

Pour les liaisons non-équilibrées :

## ■ **NRM : Normal Response Mode**

La secondaire n'émet que si elle y est invitée, et indique la fin de transmission pour rendre la main à la primaire.

→ Classe UN : Unbalanced Normal.

## ■ **ARM : Asynchronous Response Mode**

La secondaire émet à son gré (une fois la liaison activée).

→ Classe UA : Unbalanced Asynchronous.

# Modes de fonctionnement des stations

Pour les liaisons équilibrées :

- **ABM : Asynchronous Balanced Mode**

Primaire et secondaire peuvent initialiser la liaison et émettre quand elles veulent.

→ Classe BA : Balanced Asynchronous.

# Structure de la trame HDLC

Fanion	Adresse	<u>Commande</u>	Données	FCS	Fanion
01111110	(8 bits)	<b>(8 bits)</b>	(n ≥ 0 bits)	(16 bits)	01111110

FCS : Frame Check Sequence (polynôme  $X^{16} + X^{12} + X^5 + 1$ )

→ Calculs *avant* le rajout des bits de transparence à l'émission, *après* leur suppression à la réception.

Adresse : adresse d'un couple primaire/secondaire opposés

→ Dans une trame commande, adresse de la station qui reçoit.

→ Dans une trame réponse, adresse de la station qui répond.

# Le champ Commande

définit le type de la trame et ses fonctions.

**Type I ( Information) :** transfert de données.

N(R)	P/F	N(S)	0
------	-----	------	---

**Type S ( Supervision) :** accusé de réception et contrôle de flux.

N(R)	P/F	S	S	0	1
------	-----	---	---	---	---

**Type U ( Unnumbered) :** connexion, déconnexion, erreurs,

M	M	M	P/F	M	M	1	1
---	---	---	-----	---	---	---	---

N(S) : numéro trame I envoyée. N(R) : numéro trame I attendue.

P/F (Poll/Final) : P pour commandes, F pour réponses.

# Trames de supervision (1)

## ■ Trame RR (Receive Ready)

N(R)	P/F	0	0	0	1
------	-----	---	---	---	---

→ prête à recevoir

→ accusé de réception jusqu'à la trame  $N(R)-1$

# Trames de supervision (2)

## ■ Trame RNR (Receive Not Ready)

N(R)	P/F	0	1	0	1
------	-----	---	---	---	---

→ demande de suspension temporaire de toute transmission

→ accusé de réception jusqu'à la trame  $N(R)-1$

# Trames de supervision (3)

- Trame REJ (Reject) :

N(R)	P/F	1	0	0	1
------	-----	---	---	---	---

→ demande de retransmission de toutes les trames à partir de la trame N(R).

# Trames de supervision (4)

## ■ Trame SREJ (Selective Reject) :

N(R)	P/F	1	1	0	1
------	-----	---	---	---	---

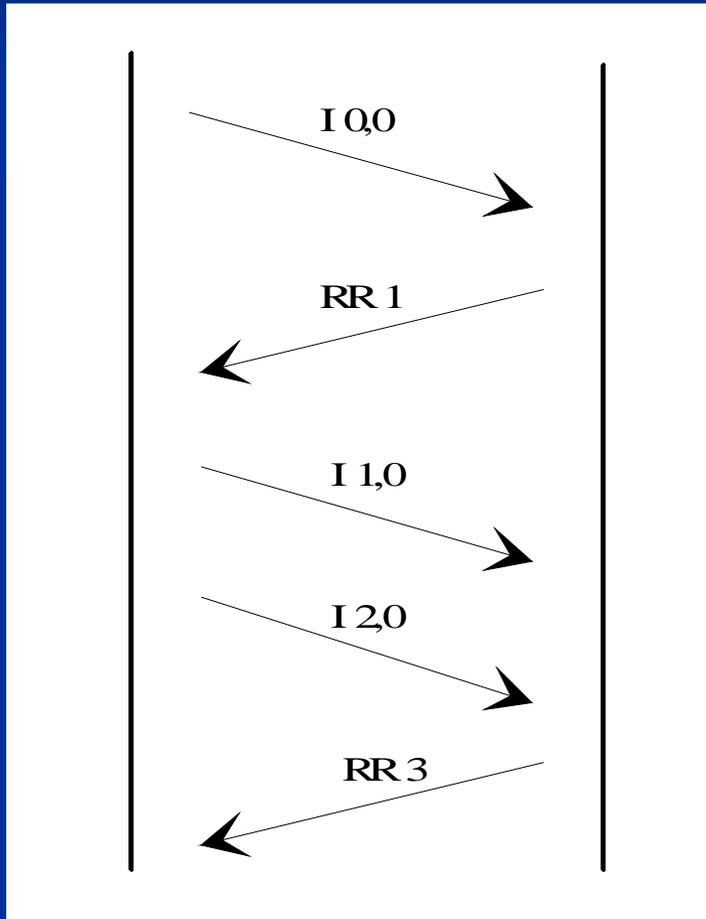
→ demande de retransmission de la trame N(R).

Ainsi, si on utilise un protocole HDLC avec :

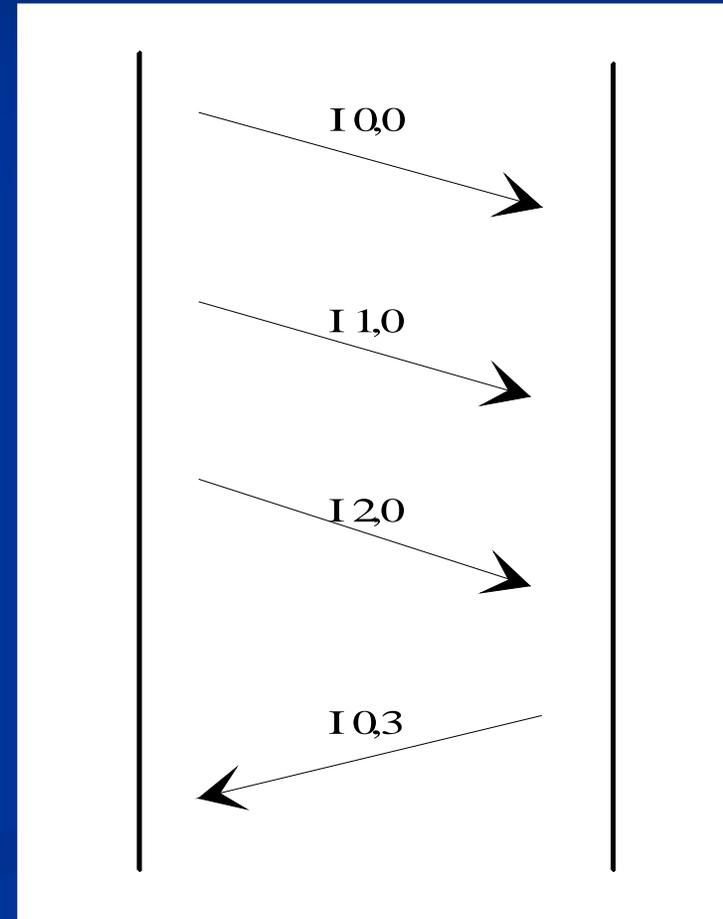
- une taille de fenêtre de réception = 1  $\Rightarrow$  REJ
- une taille de fenêtre de réception > 1  $\Rightarrow$  SREJ

# Acquittement par une

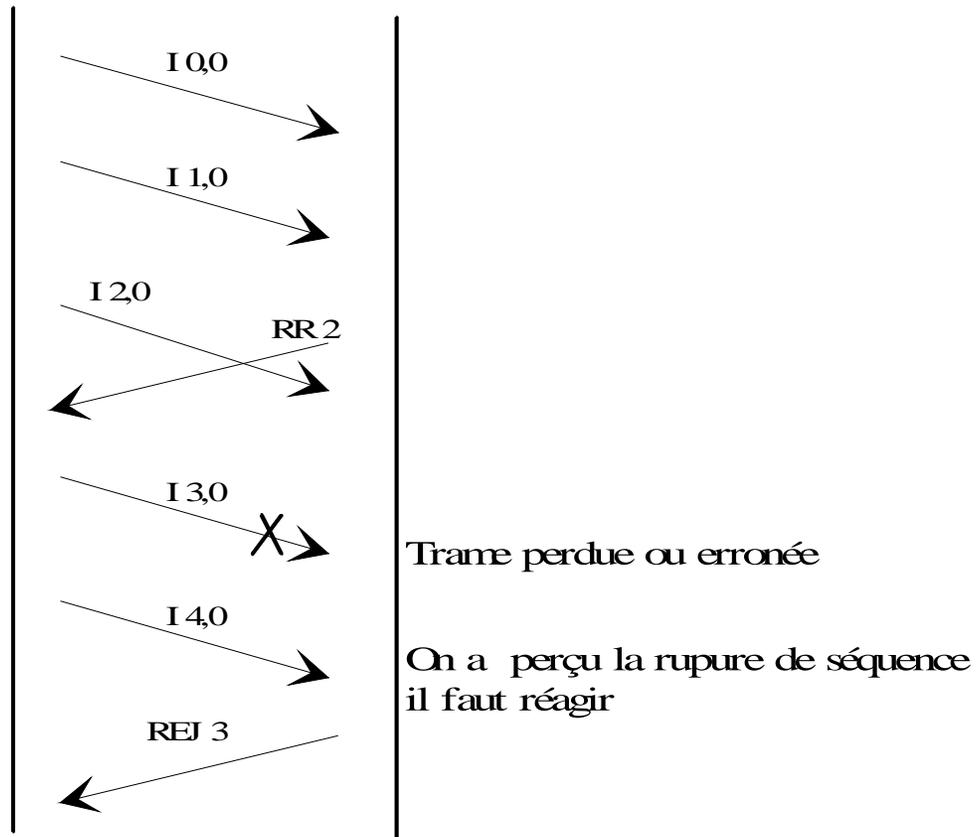
RR



I

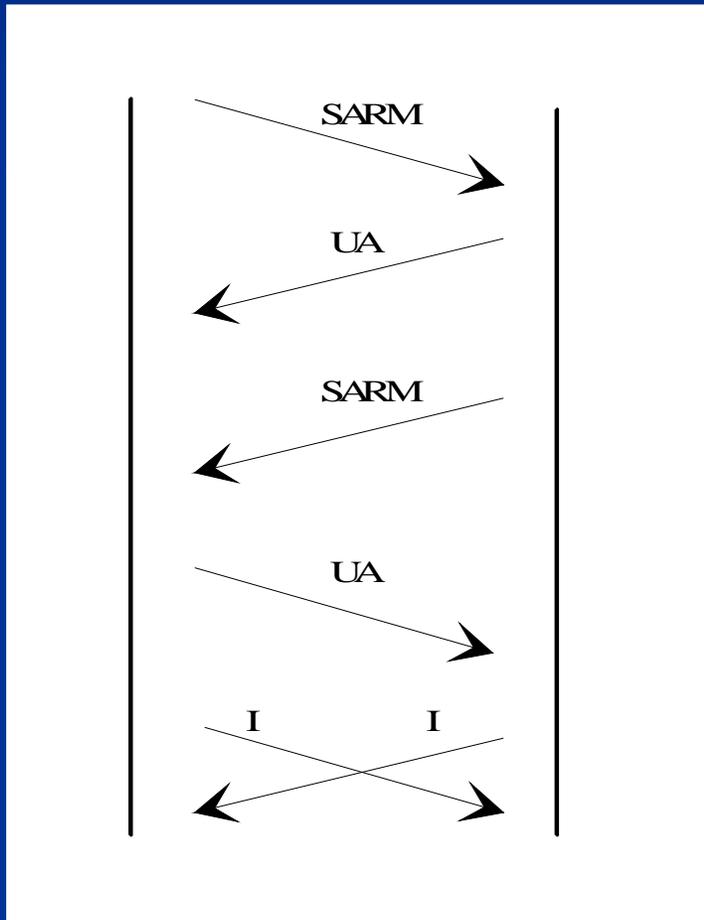


# Utilisation de REJ



# Connexion en mode

asynchrone



équilibré

