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 :

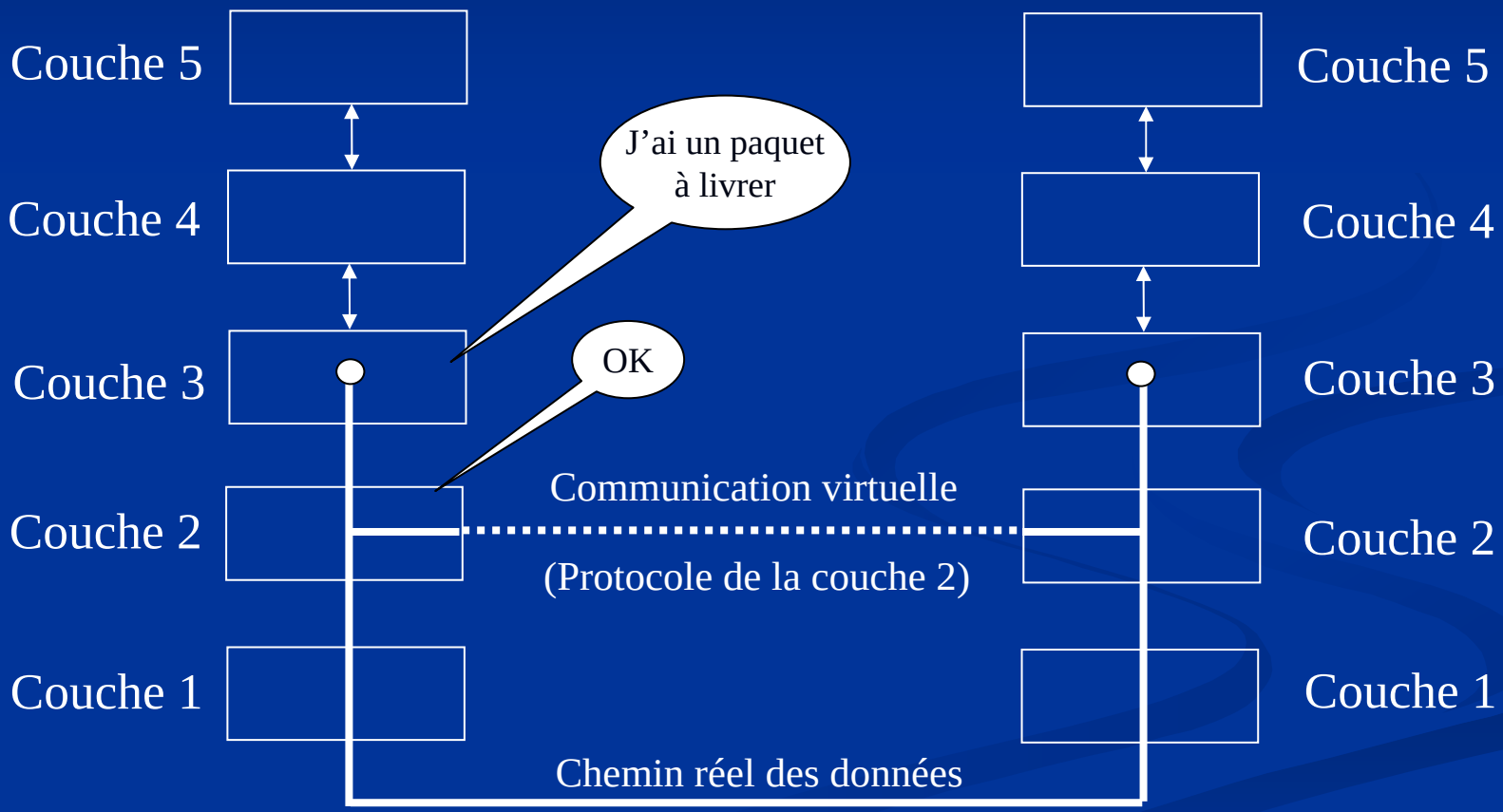
- dans l'ordre, sans erreur, sans perte , et sans duplication.

Principale fonctionnalité :

- découpage et transmission des données en trames qui sont transmises séquentiellement (confiées à la couche physique)

- **Chemin virtuel** :
  couche 2 ↔ couche 2 (trames de bits)
- **Chemin réel** :
  couche 1 ↔ couche 1 (bits sur canal de transmission)

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

Problème : un message provenant des couches hautes est découpé en 10 trames, chacune d'entre elles ayant 80% de chances d'arriver intacte. Combien de fois, en moyenne, faudra-t-il réémettre le message pour qu'il arrive entier en bon état :

- 1 : si la couche liaison n'effectue aucun contrôle d'erreur ?
- 2 : si la couche liaison effectue le contrôle d'erreur pour chaque trame ?

Problème 1

- Probabilité que n bits traversent sans erreur :

  $p = 0.8$

- Probabilité qu'un message de 10×n bits traversent sans erreur?

  $p' = p^{10} = 0.8^{10} = 0.1074$

  (1-p' = 0.8926 est donc la probabilité qu'il faudra retransmettre le message -- i.e. encore une fois les 10×n bits…)

- Nombre moyen de retransmissions

  $N = 1 / p' = 9.31$

  (plus de 9 fois, en moyenne)

Problème 2

- Combien de transmissions de n bits seraient requises *au total* si les ACK se situaient à ce niveau (et non au niveau du message complet) ?

  1- p= 0.2 est la probabilité que les n bits soient retransmis

  Pour un message de 10×n bits, on peut donc dire (en gros):
  - 8×n bits seront transmises avec succès
  - 2×n bits devront être retransmises une deuxième fois

  → au total : 12 transmissions de n bits
  (i.e. 1.2 fois le message, au lieu de 9.31 …)

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** ».

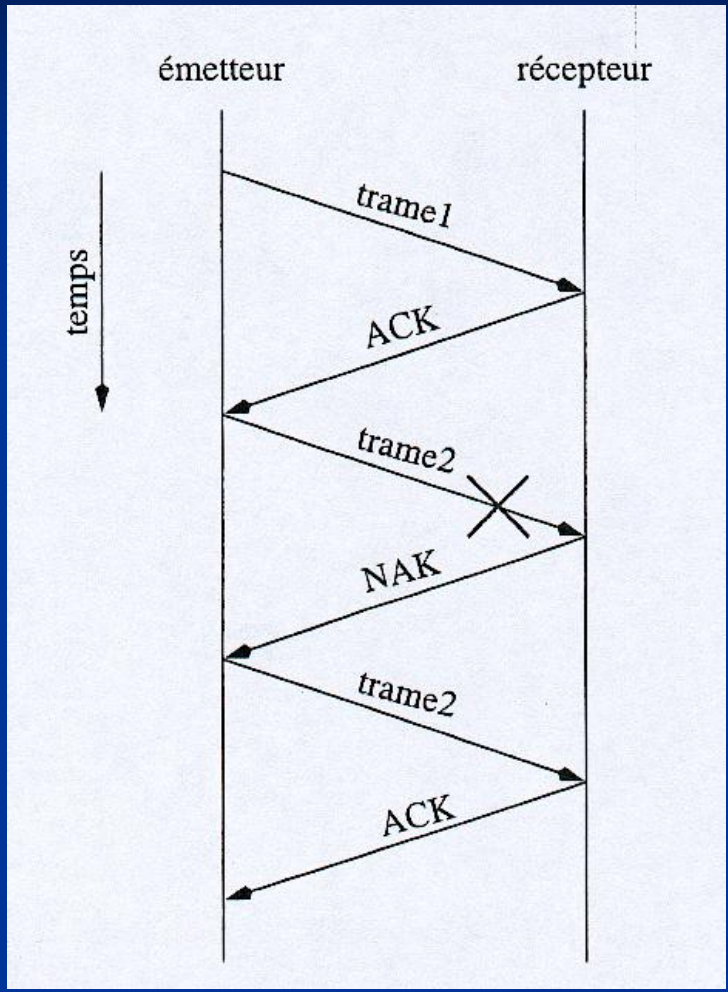
Protocoles « envoyer et attendre »

- **But** : empêcher l'émetteur d'envoyer des données plus rapidement que le récepteur ne peut les traiter.

  $\Rightarrow$ or, ralentir le débit des trames en temporisant n'est pas satisfaisant : quel choix de temporisation ?!...

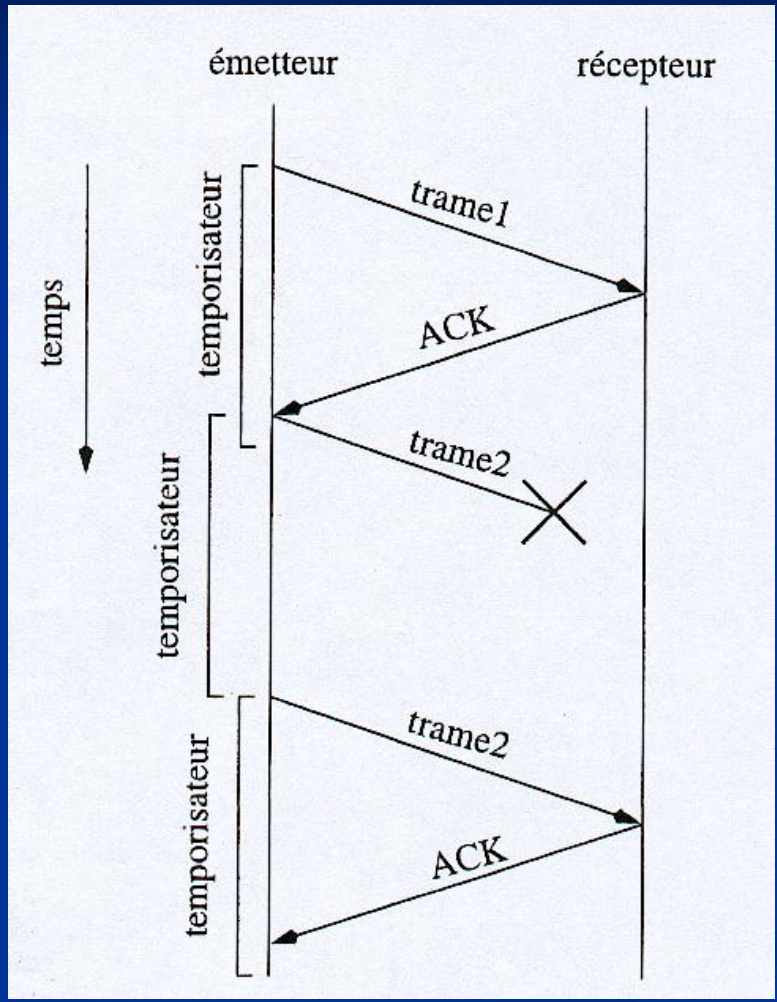
- **Méthode** :
  - Obliger le récepteur à informer l'émetteur de son état $\rightarrow$ acquittements.
  - Côté émetteur : envoyer et attendre.

Protocoles ARQ (1)



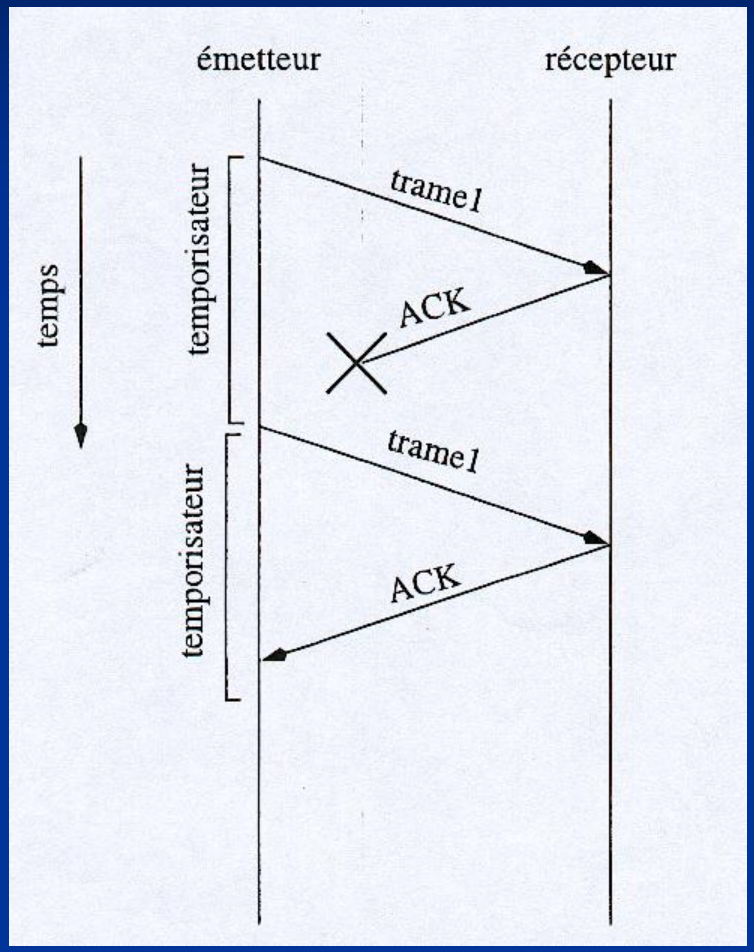
Trame erronée : acquittement positif ou négatif

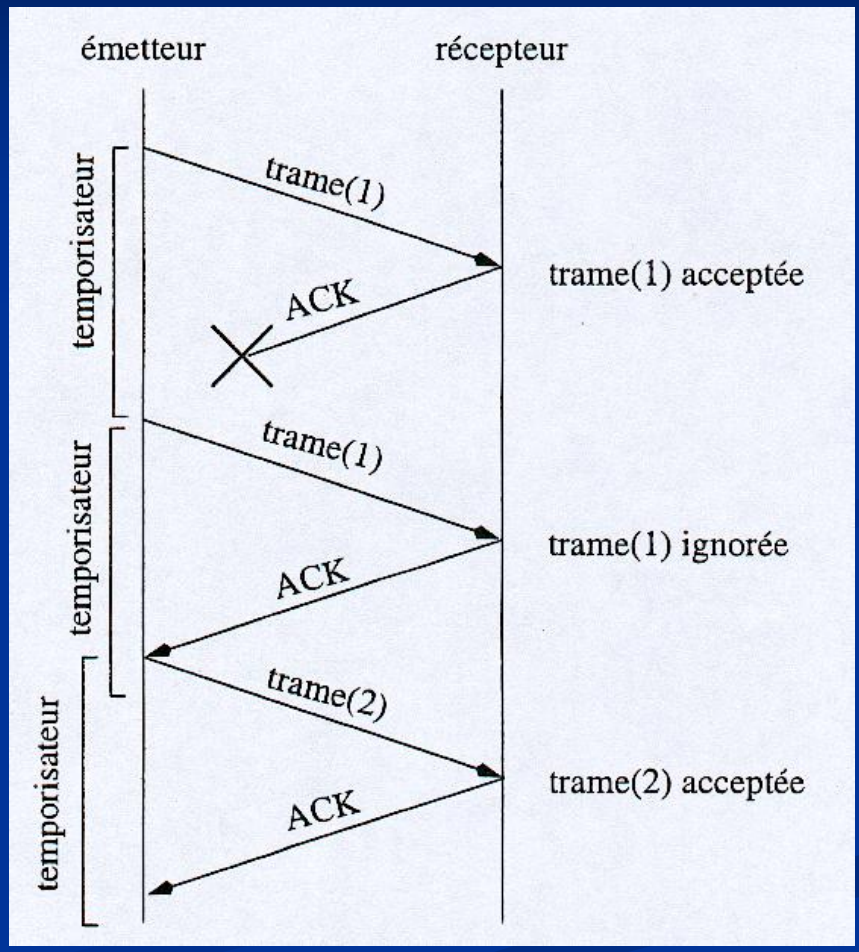
Protocoles ARQ (2)



Trame perdue : temporisateur

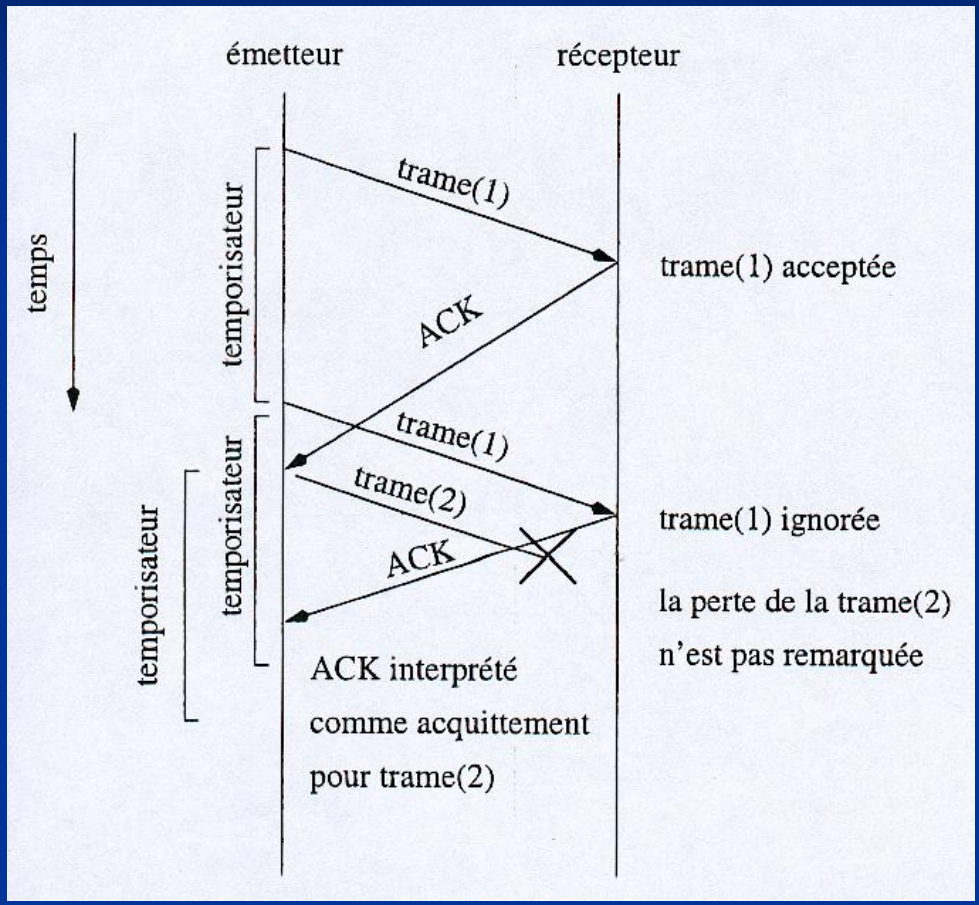
Protocoles ARQ (3)

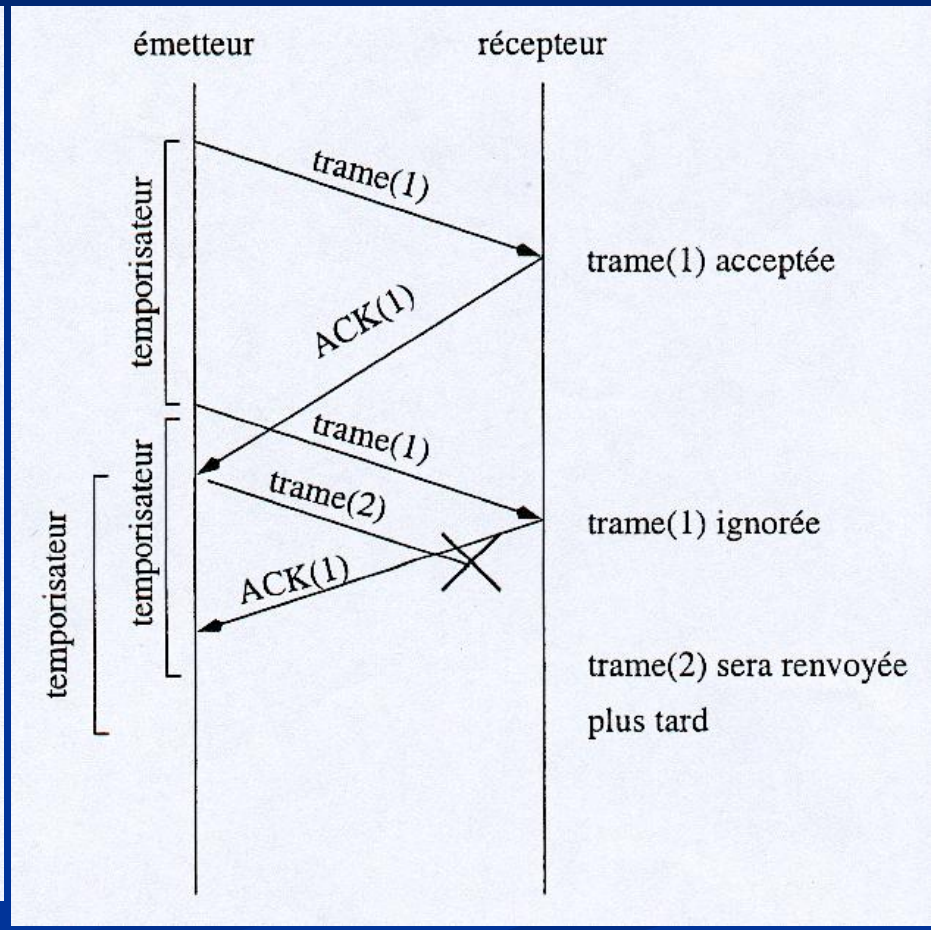




Acquittement perdu, duplication : numérotation des trames

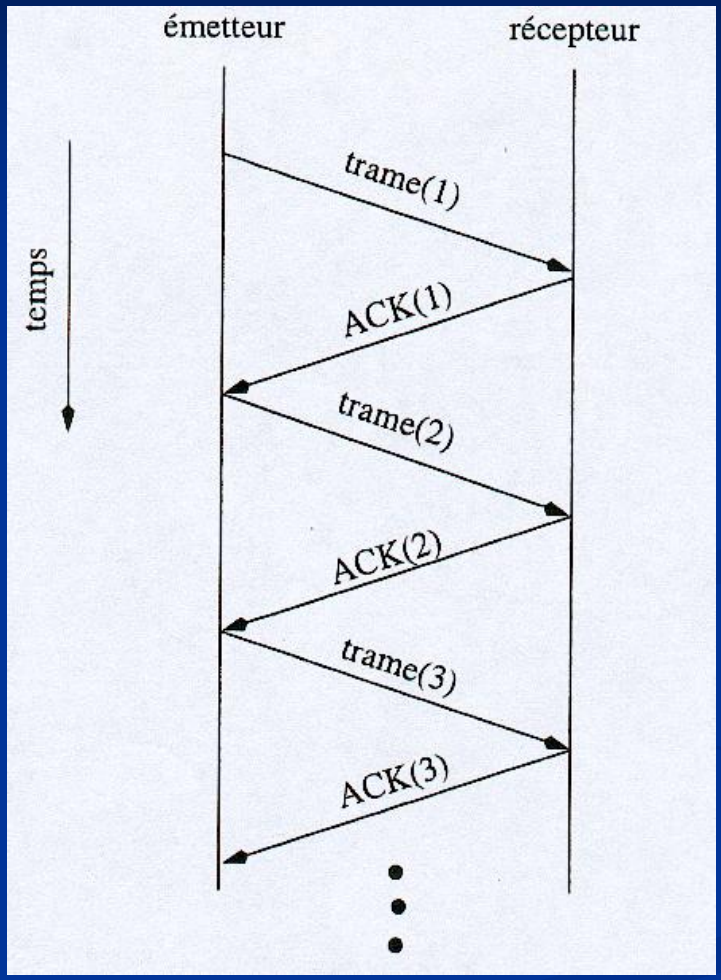
Protocoles ARQ (4)





Temporisateur expire trop tôt : numérotation des acquittements

Protocoles ARQ envoyer et attendre



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

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'acquittement (pipelining)
- Acquittements ajoutés à des trames de données envoyées dans l'autre sens (piggypacking).

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

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

Fenêtre d'anticipation

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

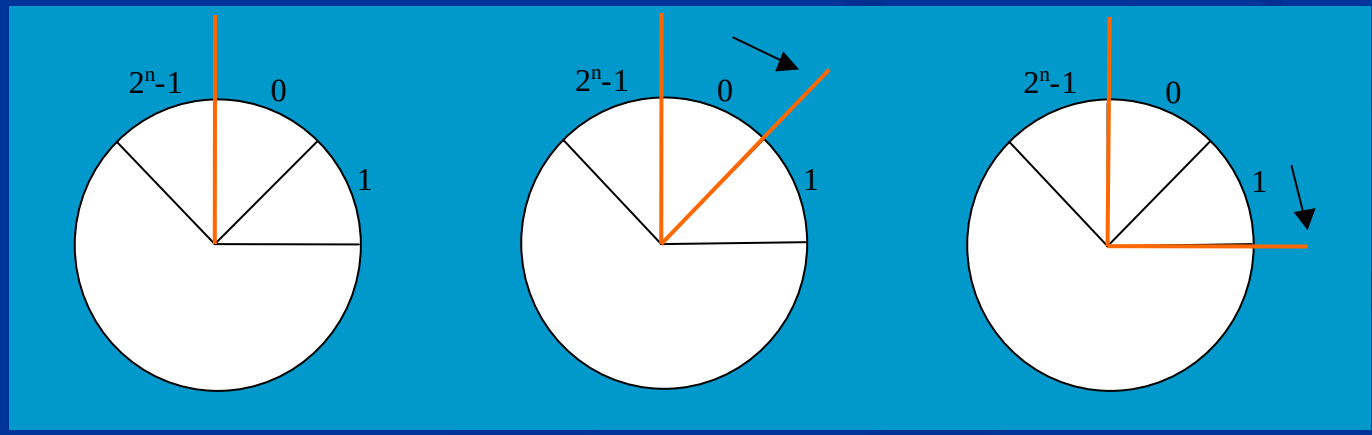
Remarque : si n=1 $\Rightarrow$ « envoyer et attendre » robuste.

- **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* .

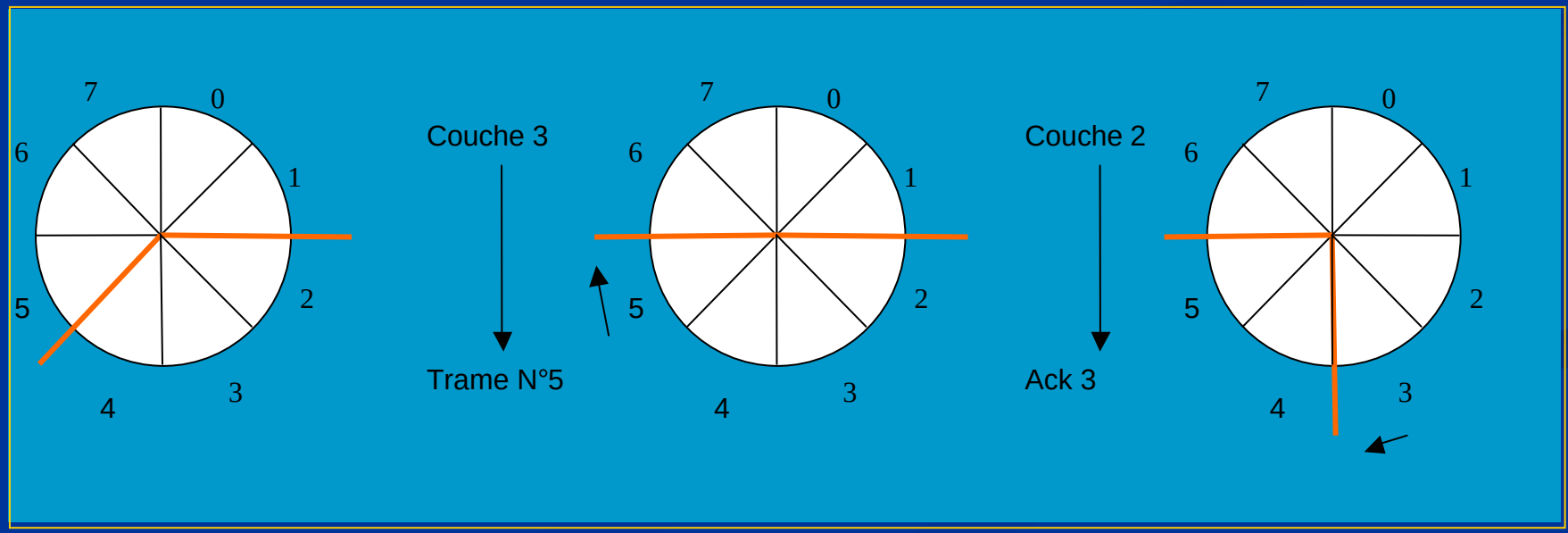
Fenêtre d'émission

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

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



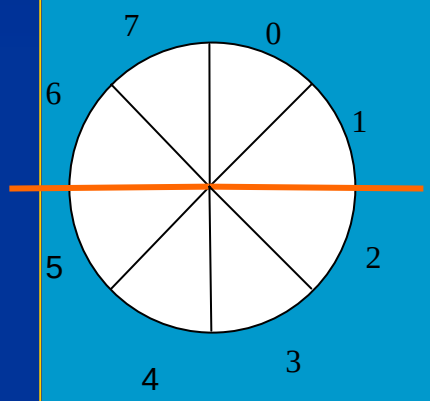
- L'émetteur stocke les trames non acquittées dans des zones tampons (au plus *m* trames, si *m* est la taille de la fenêtre).
- Si la fenêtre atteint son maximum, on n'envoie plus rien jusqu'à une libération, ie un acquittement.



Fenêtre de réception

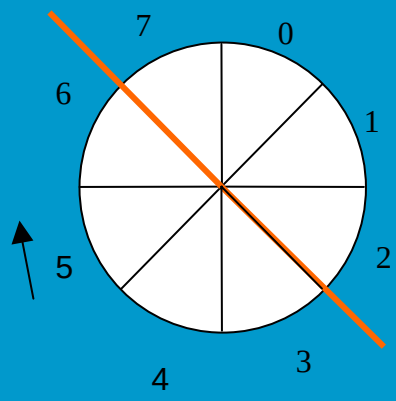
Taille (fixe) = nombre de trames autorisées à être reçues.

Contenu = numéros de séquence de ces trames attendues.



Couche 1

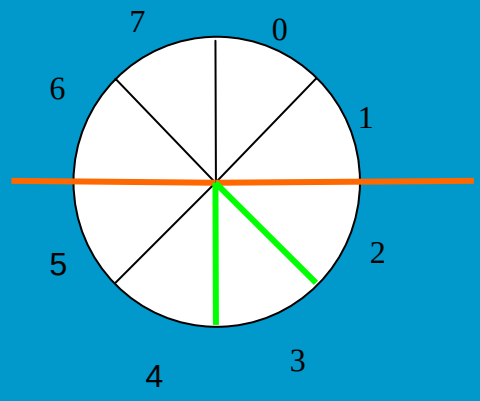
Trame N°2



Trame N° 2 vers couche 3.

Couche 1

Trame N°3



Trame N° 3 stockée.

Exemples de protocoles (1)

- **Protocole à fenêtres d'émission et de réception de largeur 1**

→ revient à « envoyer et attendre » robuste.

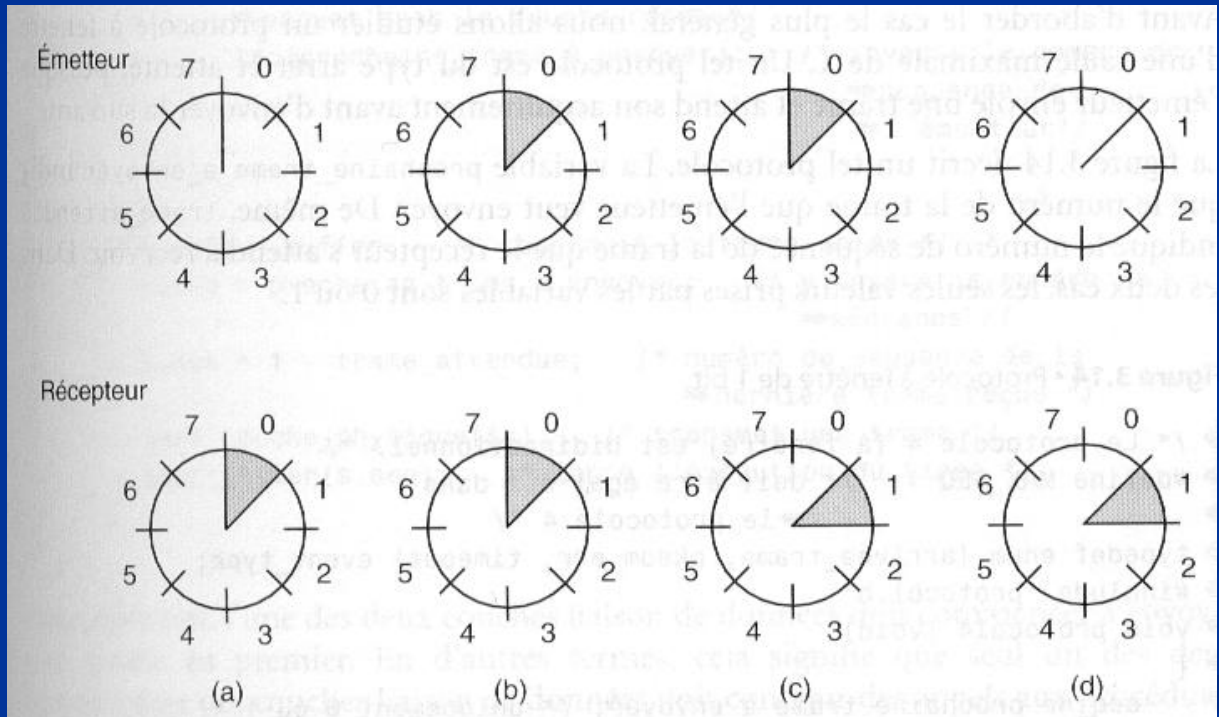


Figure 3.13 • Fenêtre de taille 1, avec un numéro de séquence sur 3 bits. (a) Initialement. (b) Après que la première trame a été envoyée. (c) Après que la première trame a été reçue. (d) Après que le premier acquittement a été reçu.

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.

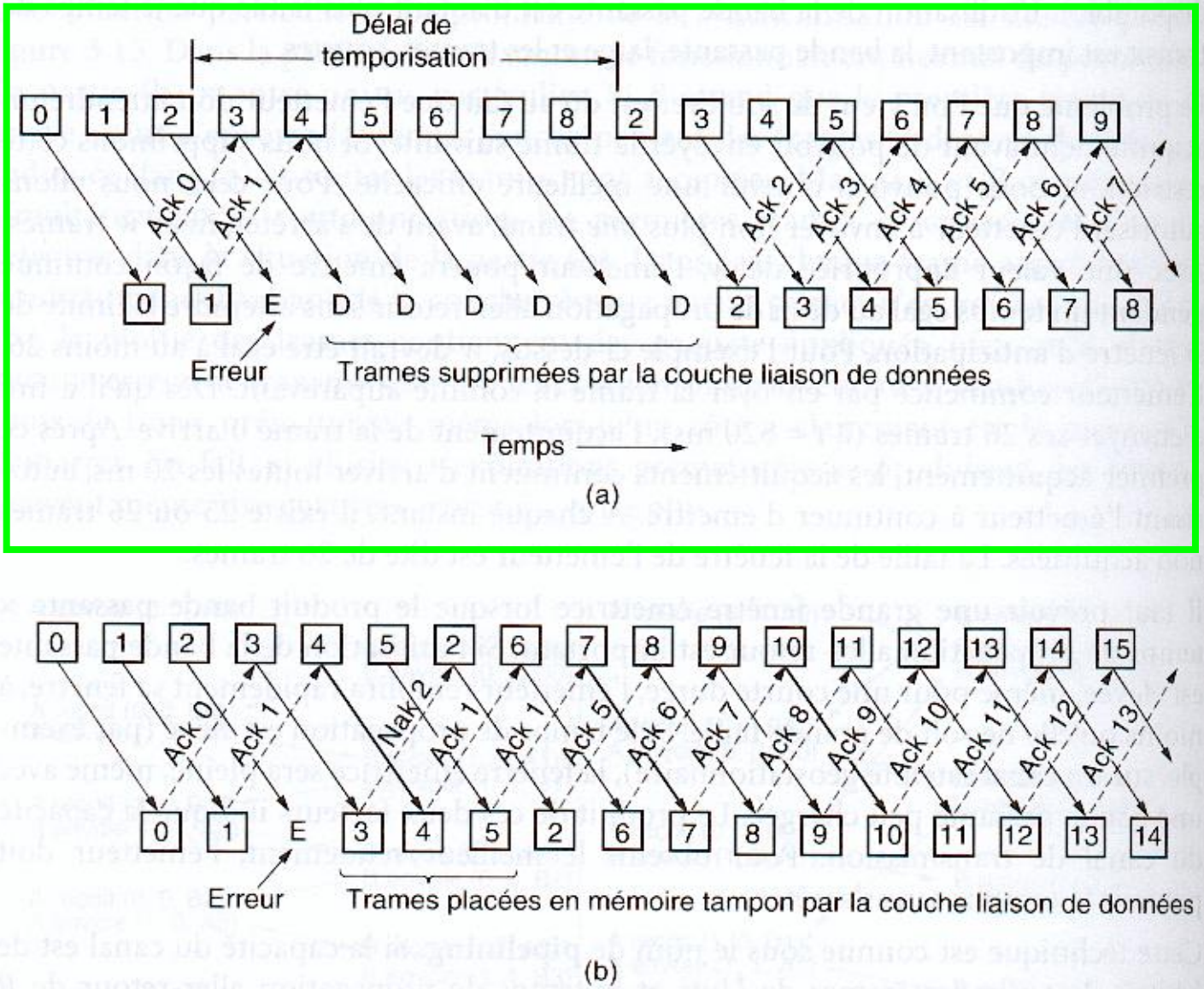


Figure 3.16 • Pipelining et récupération d'erreur. Effet d'une erreur lorsque (a) la fenêtre du destinataire a une taille de 1 et (b) lorsqu'elle est de taille importante.

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.

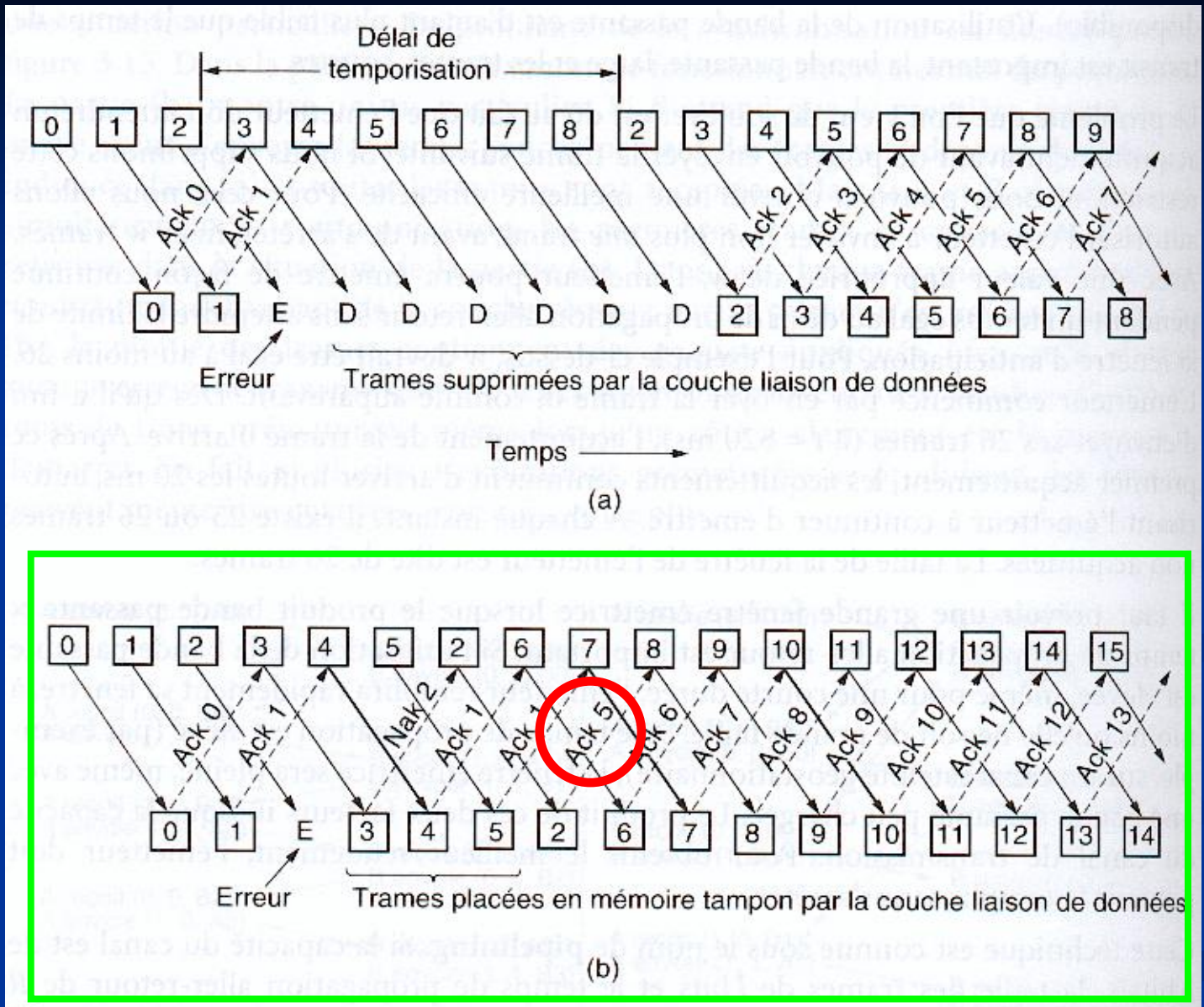


Figure 3.16 • Pipelining et récupération d'erreur. Effet d'une erreur lorsque (a) la fenêtre du destinataire a une taille de 1 et (b) lorsqu'elle est de taille importante.

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 $t_e <= 2^n - 1$,
  - 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	**Commande**	Données	FCS	Fanion
01111110	(8 bits)	**(8 bits)**	(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 (**I**nformation) : transfert de données.

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

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

N(R)	P/F	S	S	0	1

Type U (**U**nnumbered) : 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

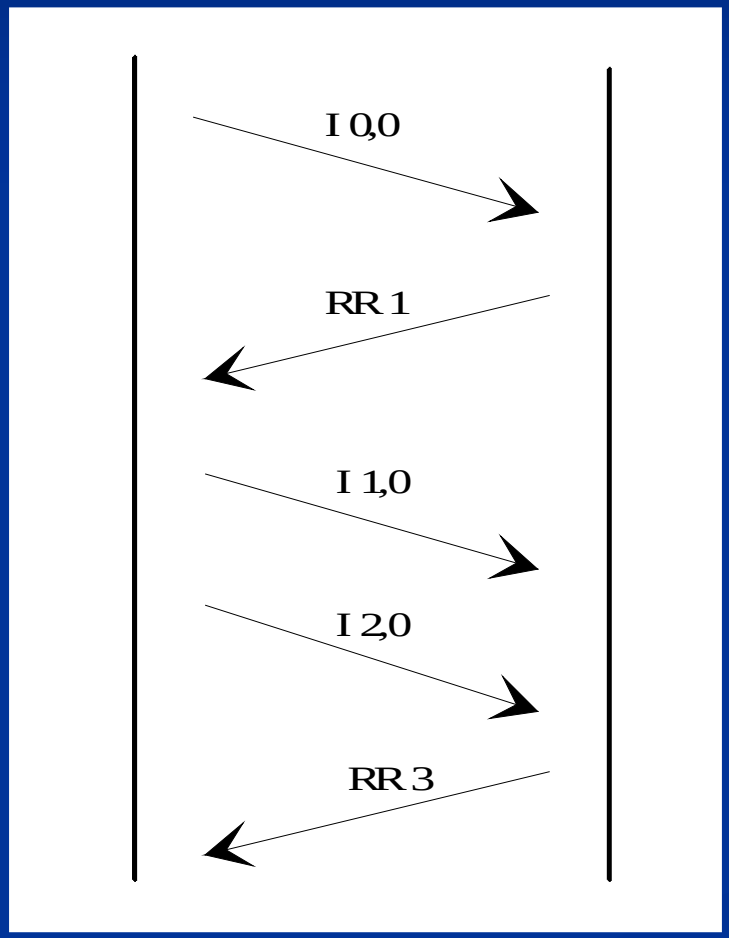
→ demande de retransmission de la trame N(R).

Ainsi, si on utilise un protocole HDLC avec :

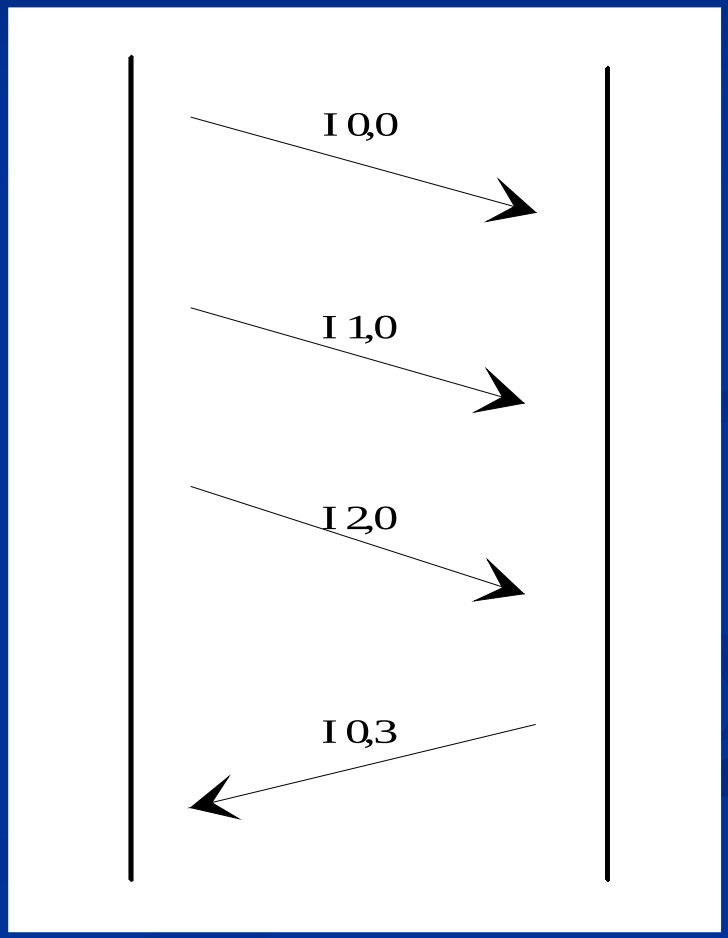
- une taille de fenêtre de réception =1 ⇒ REJ
- une taille de fenêtre de réception >1 ⇒ SREJ

Acquittement par une

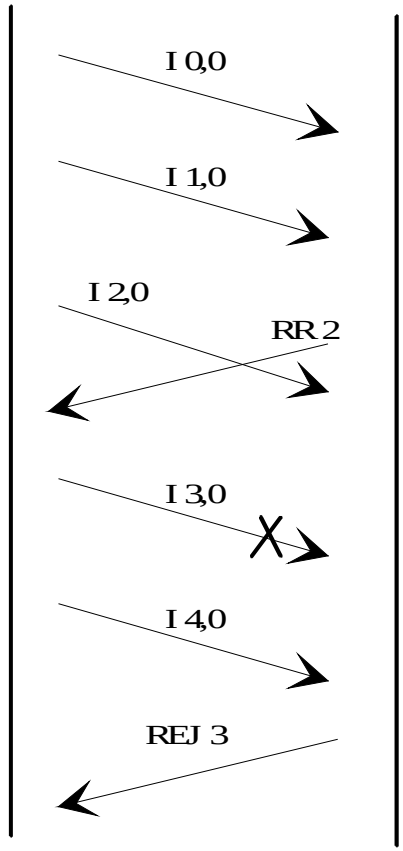
RR



I



Utilisation de REJ

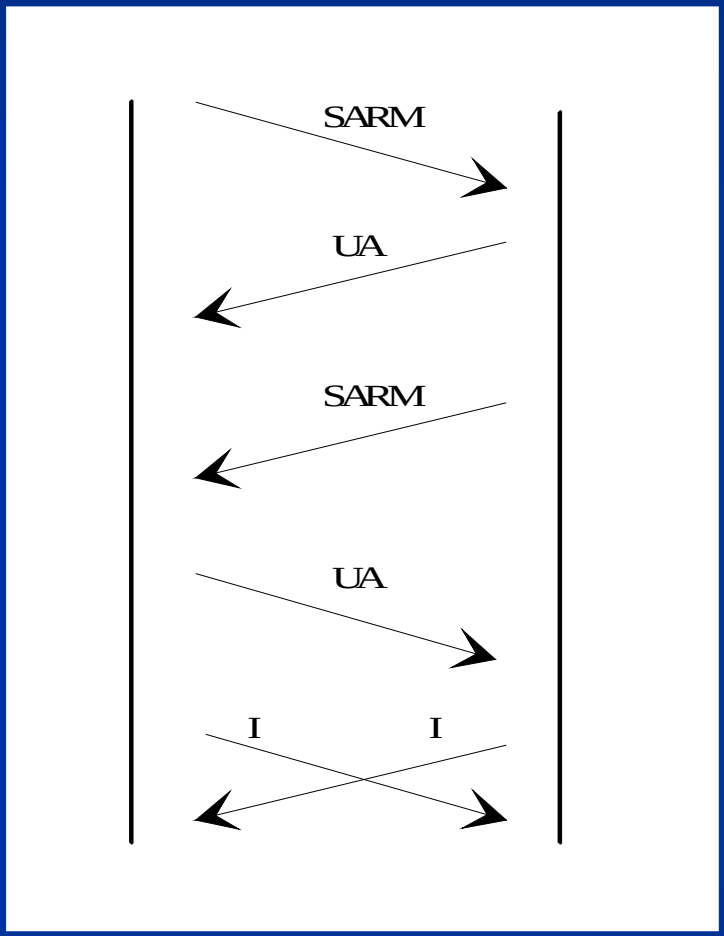


Trame perdue ou erronée

On a perçu la rupure de séquence il faut réagir

Connexion en mode

asynchrone



équilibré