Couche de Liaison

IFT3320

A. S. Hafid
e-mail: ahafid@iro.umontreal.ca
phone: (514) 343-2446

Plan

- Couche Liaison: Objective
- Trames
- Détection et correction des erreurs
- Contrôle de flux et d'erreurs
- HDLC
- PPP
- Conclusion

Couche Liaison

- Objectifs de la couche liaison de données
  - fournir un transfert fiable et efficace de paquets à son "utilisateur“ (réseau) sachant que
    - la couche physique transmet des bits et pas de paquets
    - la couche physique peut subir des erreurs de transmission
      - erreurs de transmission
      - perte de bits
      - création de bits
- Les fonctions principales des protocoles de la couche liaison sont:
  - contrôle d’erreurs de transmission
  - contrôle du flux, gestion de la liaison
- Le rôle de la couche liaison est de faire paraitre un canal de communication **non-fiable** comme étant **parfait**

Couche Liaison

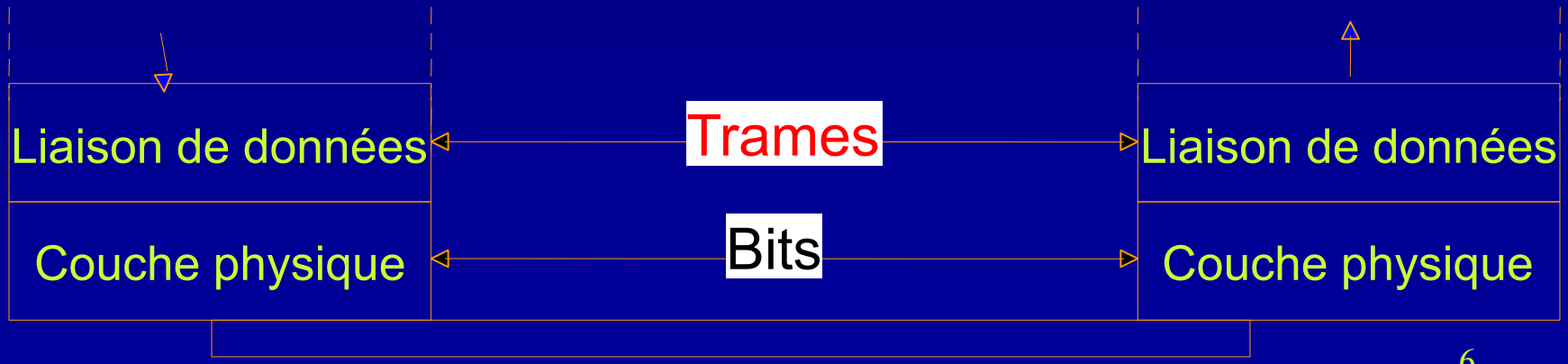
- La couche physique transmet des bits
  - Qu'est-ce qu'il faut faire, en premier, pour pouvoir faire le contrôle d'erreurs?

Couche Liaison

- Qu'est-ce qu'il faut faire, en premier, pour pouvoir faire le contrôle d'erreurs?
  - Notion d'une unité de données
    - Trame

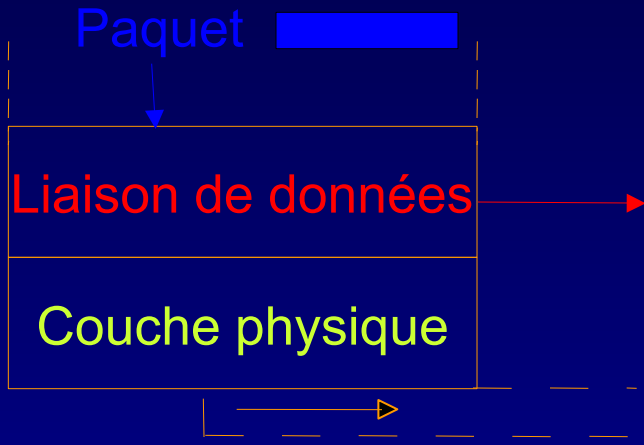
Trames

- Trame
  - unité d'information transférée entre deux entités de la couche liaison de données
  - groupe logique de *N* bits
  - La taille des trames est en général variable



Trames (Cont.)

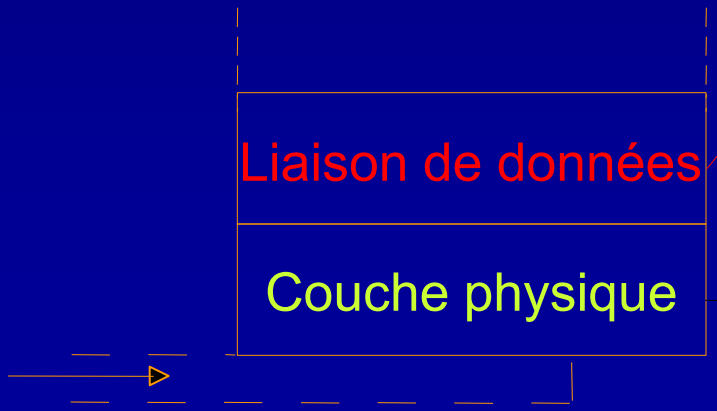
- l'émetteur



Reçoit un paquet à transmettre, le place dans une trame et confie la suite de bits correspondant à la couche physique pour transmission

Entête ajoutée par la couche 2

- receveur



Reçoit en permanence un flot continu de bits.
Comment identifier chaque trame dans ce flot?

Reçoit en permanence un flot continu de signau électriques/optiques qu'elle délivre comme bits à la couche liaison de données

Problème

- Comment reconnaitre une trame?
  - Suite de bits

Trames: Character Stuffing

- Marquer à la fois le début et la fin de la trame
- Character stuffing
  - Une solution parmi d'autres
  - 'DLE' 'STX' en début et 'DLE' 'ETX' en fin
- Si le caractère 'DLE' apparaît à l'intérieur de la trame, quoi faire?

Trames: Character Stuffing

- L'émetteur remplace 'DLE' par 'DLE' 'DLE' si le caractère 'DLE' apparaît à l'intérieur de la trame
- Exemple
  - Paquet : 1 2 3 'DLE' 4
  - Trame construite par la couche liaison de données'DLE' 'STX' 1 2 3 'DLE' 'DLE' 4 'DLE' 'ETX'

Trames: Character Stuffing (Cont.)

- Paquet 1 : 1 2 3 'DLE' 4
  - Trame 1 : 'DLE' 'STX' 1 2 3 'DLE' 'DLE' 4 'DLE' 'ETX'
- Paquet 2 : 'DLE' 'STX' 'DLE' 'ETX'
  - Trame 2 : 'DLE' 'STX' 'DLE' 'DLE' 'STX' 'DLE' 'DLE' 'ETX' 'DLE' 'ETX'
- Paquet 3 : 'STX' 'DLE'
  - Trame 3 : 'DLE' 'STX' 'STX' 'DLE' 'DLE' 'DLE' 'ETX'
- Chaîne d'octets reçue par le receveur 'DLE' 'STX' 1 2 3 'DLE' 'DLE' 4 'DLE' 'ETX' 'DLE' 'STX' 'DLE' 'DLE' 'STX' 'DLE' 'DLE' 'ETX' 'DLE' 'ETX' 'DLE' 'STX' 'STX' 'DLE' 'DLE' 'DLE' 'ETX'

Trames: Bit Stuffing

- Une alternative à character stuffing
  - 01111110 en début et en fin de trame
- Comment faire si cette suite se répète?

Trames: Bit Stuffing (cont.)

- Si cinq bits à '1' apparaissent consécutivement dans un paquet à transmettre, l'émetteur insère un bit à '0' après le cinquième bit à '1'
- Exemple
  - Paquet : 011011111111111111110010
  - Trame construite par la couche liaison de données 01111110011011111011111011111011001001111110

Trames: Bit Stuffing (cont.)

- Avantage par rapport à character stuffing
  - Une trame ne doit pas nécessairement contenir un nombre entier d'octets

Trames

- En pratique
  - Character stuffing ou bit stuffing
  - code de contrôle pour vérifier l'absence d'erreurs de transmission dans la trame
- Une trame reçue est considérée valide si
  - le bon délimiteur est au début
  - le bon délimiteur est à la fin
  - le code de contrôle est correct

Couche Liaison

- Etablir, maintenir et arrêter la communication entre nœuds adjacents
- L'unité d'information transmise est la trame (frame…).
- Les bits sont reçus dans l'ordre où ils ont été envoyés.
- Problèmes potentiels :
    - trame abîmée
    - trame perdue
    - récepteur ne suit pas

Transmission Fiable

- Méthodes pour la transmission fiable?

Transmission Fiable

- On distingue trois classes de méthodes pour la transmission fiable
  - **Écho:** détection d'erreurs par l'émetteur suivie d'une retransmission.
  - **Codes correcteurs:** détection et correction d'erreurs par le récepteur.
  - **Détection et retransmission:** détection d'erreurs par le récepteur suivie d'une retransmission
    - Protocoles du type ARQ (*Automatic Request Repeat*).
    - Il existe deux types de protocoles de liaison ARQ:
      - » protocoles *envoyer et attendre* (stop-and-wait)
      - » protocoles *continues* (continuous ARQ ou pipelined ARQ)

Détection et correction des erreurs

- Problème
  - La couche physique n'est pas parfaite
- Types d'erreurs
  - Erreur simple : un bit est inversé dans la trame
  - Erreurs en rafale : un groupe de N bits à l'intérieur de la trame sont affectés par une erreur
    - Exemple
    - 01111110011011111011111011111011001001111110
    - 01111110011011111*01011110*11101100100111110
      - longueur de la rafale : 8 bits (le premier et le dernier bits de la rafale sont erronés, mais à l'intérieur de la rafale certains bits peuvent être corrects)
  - Les erreurs les plus courantes?

Détection et correction des erreurs

- Les erreurs en rafales sont plus courantes que les erreurs simples

Correction des erreurs

- Code correcteur d'erreur
  - Idée : transmettre de l'information redondante pour permettre au receveur de corriger des erreurs
- Exemple :
  - transmettre '111' ('000') à la couche physique pour chaque '1' ('0') à l'intérieur d'une trame
  - On suppose l'existence d'une erreur simple ou non; dans ce cas
    - Si on reçoit 111000, ça correspond à quelle suite de bits envoyée?
    - Si on reçoit 111111, ça correspond à quelle suite de bits envoyée?
    - Si on reçoit 111101, ça correspond à quelle suite de bits envoyée?
    - Si on reçoit 100111, ça correspond à quelle suite de bits envoyée?
  - On suppose l'existence d'une erreur double ou non; dans ce cas
    - Si on reçoit 111000, ça correspond à quelle suite de bits envoyée?
    - Si on reçoit 111111, ça correspond à quelle suite de bits envoyée?
    - Si on reçoit 111101, ça correspond à quelle suite de bits envoyée?
    - Si on reçoit 100111, ça correspond à quelle suite de bits envoyée?

Correction des erreurs

- En cas d'erreurs simples, '111' devient
  - '011' ou '101' ou '110'
- En cas d'erreurs simples, '000' devient
    - ‘1 00' ou '010' ou '001'
- Dans tous ces cas d'erreurs simples, le receveur peut détecter le message original transmis
  - mais pas en cas d'erreurs doubles

Correction des erreurs (Cont.)

- Principe
  - Considérons un message de *m* bits de données
  - Transmettre *m*+*r* bits au lieu de *m* bits et utiliser les *r* bits pour corriger les erreurs de transmission
  - Exemple
    - code '111' et '000', *m*=1 et *r*=2
  - Seuls 2^m parmi les $2^{(m+r)}$ séquences de bits correspondent à des codes valides
    - Exemple: 011, 101, 110, 100, 010, 001, 111, 000
  - Les 2^m séquences de bits transmises = les codewords
    - Exemple: 111 et 000

Distance de Hamming et Encodage

- Avec de mots de *m* bits, on a 2^m mots code valides
  - Les *m* bits du mot déterminent les *r* bits de validation
    - » On a *r* = encodage(*m*)
  - On a seulement 2^m mots codes valides parmi les 2^n mots code possibles
- On a une erreur si un mot code invalide est détecté
- La distance de Hamming d'un encodage
  - La plus petite distance de Hamming entre deux mots code
  - Exemple :
    - code '111' et '000', distance de Hamming = 3
- Détection et auto-correction d'un encodage
  - Avec une distance de *d*, on peut détecter *d*-1 erreurs de 1 bit
  - Avec une distance de *d*, on peut corriger (*d*-1)/2 erreurs de 1 bit

Distance de Hamming et Encodage: Exemple

- On propose le code (4 mots valides) suivant 000000, 000111, 111000, et 111111
- Calculer le nombre d'erreurs simples qu'on peut détecter et le nombre d'erreurs simples qu'on peut corriger.

Parité (1)

- On ajoute un bit au mot pour créer un mot code (r = 1)
  - Parité paire
    - Le bit de parité est 0 si le nombre de bits 1 est pair, sinon il est 1.
    - 011101000
    - 011111001
  - Parité impaire
    - Le bit de parité est 1 si le nombre de bits 1 est pair, sinon il est 0.
- Cet encodage a une distance de Hamming *d ?*

Parité (2)

- Cet encodage a une distance de Hamming $d = 2$
  - La première erreur rend la parité erronée
  - La deuxième erreur correspond à un mot code valide
- Exemples
  - message de m bits : 10110101
  - avec parité paire : 101101011
  - avec parité paire : 001101010

Parité (3)

- Une utilisation simple pour corriger une erreur simple et détecter une erreur double?

Code de Parité Bidimensionnelle (1)

- Code de Parité Bidimensionnelle (Parité Horizontale et Verticale)
  - Le concept consiste à créer une matrice avec les bits de données, et à ajouter :
    - Un bit de parité pour chaque ligne (parité horizontale).
    - Un bit de parité pour chaque colonne (parité verticale).
    - Un bit de parité global qui couvre toute la matrice

Code de Parité Bidimensionnelle (2)

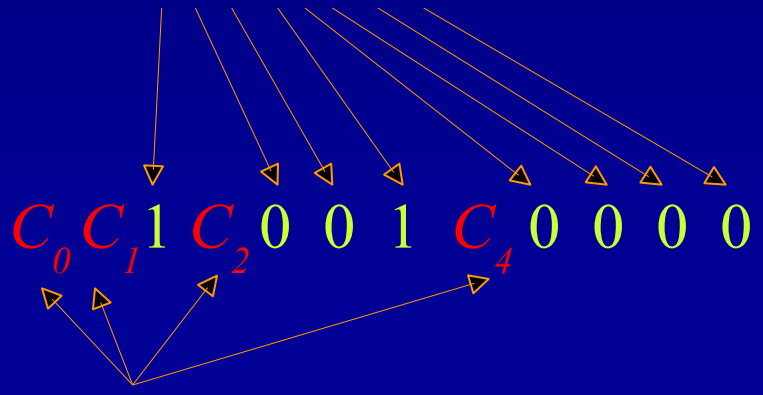
- 1 0 1 1 | 1 (parité de la 1ère ligne)
- 0 1 0 1 | 0 (parité de la 2ème ligne)
- 1 1 1 0 | 1 (parité de la 3ème ligne)
- 0 0 1 0 | 1 (parité de la 4ème ligne)
- ----------
- 0 0 1 0 | 1 (parité de chaque colonne et parité globale)

Code de Hamming

- Code optimal pour détecter/corriger les erreurs simples
- Extension du mécanisme de parité
- Principe
  - Numéroter séquentiellement les *m* bits à transmettre en commençant à gauche
  - Les bits 1,2,4,8,16,32,... sont des bits de contrôle
  - Les bits 3,5,6,7,9,10,11,... servent aux données
- Calcul des bits de contrôle
  - un bit de contrôle est la parité (paire ou impaire) calculée sur un groupe de bits de données bien choisi

Code de Hamming (Cont.)

- Exemple
  - transmission d'octets
  - bits de contrôle en position 1,2,4,8
  - bits de données en position 3,5,6,7,9,10,11,12
  - le bit de donnée d'indice *i* intervient dans le calcul des bits de contrôle dont les indices interviennent dans la décomposition binaire de *i*
  - Exemple
    - message à transmettre 10 0 1 00 0 0
    - Code de Hamming :



Bits calculés

Code de Hamming (Cont.)

$C_0\ C_1\ 1\ C_2\ 0\ 0\ 1\ C_4\ 0\ 0\ 0\ 0$

- Calcul des bits de contrôle
  - Bits de données (décomposition binaire des indices)
    - 3=1+2, 5=1+4, 6=2+4,7=1+2+4,9=1+8, 10=2+8, 11=1+2+8, 12=4+8
  - Bits de contrôle
    - Bit 1 : parité paire sur les bits 3,5,7,9,11 ->0 C0
    - Bit 2 : parité paire sur les bits 3,6,7,10,11 ->0 C1
    - Bit 4 : parité paire sur les bits 5,6,7,12 -> 1 C2
    - Bit 8 : parité paire sur les bits 9,10,11,12 -> 0 C4
- Receveur
  - Vérifier les bits de contrôle
    - Erreur simple, si bits i,j et k sont faux, erreur dans bit i+j+k

Code de Hamming (Cont.)

- C’est quoi la suite des bits (données) envoyée par la source?

0 0 1 1 0 0 1 0 0 1 0 0

Détection des erreurs

- Complexité des codes correcteurs d'erreur
  - utilisé surtout sur les canaux simplexes
- En pratique, on préfère détecter les trames erronées et les retransmettre si nécessaire
- Détection des trames erronées
  - Parité
    - Désavantage : ne détecte que les erreurs simples
  - Somme de contrôle
    - Exemple : checksum sur 16 bits

Cyclical Redundancy Code (CRC)

- Permet la détection de la grande majorité des erreurs
- Le principe de fonctionnement
  - On groupe les données à transmettre par bloc
  - On divise le bloc par un polynôme générateur (suite de bits)
  - On transmet le bloc de données concaténé avec le reste de la division
  - On vérifie la validité du bloc de données en effectuant la division des données concaténées par le polynôme générateur
    - » Les données sont valides si le reste de cette division est 0

Suite binaire vs. polynôme

- 1001001
  - x^6+x^3+1
- 1011
  - x^3+x+1
- *1111*
  - *?*
- *10000*
  - *?*

Cyclical Redundancy Code (Cont.)

Division : exemple (- : xor)

x^6+x^3+1 1001001 1011 x^3+x+1

- 1011 1010 x^3+x

0100

-0000

1000

-1011

0111

-0000

111 : Reste

x^2+x+1

Vérification : 1011*1010+111=1001001 ?

$(x^3+x+1)*(x^3+x)+x^2+x+1 = x^6+x^3+1$?

Cyclical Redundancy Code (Cont.)

- Principe
  - Calculer la somme de contrôle en arithmétique polynomiale
  - L'émetteur et le receveur se mettent d'accord sur un polynôme générateur G(x) de la forme $1*x^k+...+1*x^0$
  - Une trame de *m* bits correspond à un polynôme M(x)
  - La somme de contrôle (S(x)) est ajoutée à la fin de la trame et est calculée de façon à ce que le polynôme M(x)S(x) soit divisible par G(x)

Cyclical Redundancy Code (Cont.)

- Algorithme
  - Si *r* est le degré de G(x), ajouter *r* zéros à la suite de la trame
    - Exemple : 1011*0000*
    - La trame allongée correspond au polynôme $x^r*M(x)$
  - Diviser $x^r*M(x)$ par G(x) avec la division polynomiale
  - Si R(x) est le reste de la division, la trame à transmettre est $T(x) = x^r*M(x) - R(x)$
  - T(x) ainsi construit est divisible par G(x)
    - Rappel : si D=résultat(A/B) et R=reste(A/B), alors A=B*D+R et (A-R) est divisible par B
  - Pour vérifier qu'une trame a été reçue correctement, le receveur devra simplement vérifier que le reste de la division de T(x) par G(x) vaut zéro

Cyclical Redundancy Code (Cont.)

M(x) = 10110011, G(x)=10011, r=4

```
10110011 0000        | 10011
-10011               | 10101100
 01010
-  00000
  10101
 -  10011
   01101
  -  00000
    11010
   -  10011
     10010
    -  10011
       0001100=R(x)
```

T(x)=10110011 *0100*

Cyclical Redundancy Code: Exemple

- Un émetteur envoie une suite de bits en utilisant le polynôme générateur G(X)=10011
- Le récepteur reçoit la suite de bits suivante R(X)=1010000110100.
- Est-ce que les données reçues (R(X)) sont correctes ou erronées?

Types d'erreurs Détectées

- Supposons qu'il y a eu des erreurs durant la transmission de T(X)
  - RT(X) est reçu
- RT(X)=T(X)+E(X)
  - E(X) représente les erreurs; chaque bit 1 dans E(X) représente une erreur d'un bit
  - Une erreur en rafale est caractérisée par un premier bit 1, une mixture de 0s et 1s, et un bit final 1 , avec tous les autres bits à 0
- Exemple
  - T(X)= 101100110100
  - RT(X)= 111100010100
  - E(X)= 010000100000

Types d'erreurs Détectées (cont.)

- Le receveur divise (T(X)+E(X)) par G(X)
  - T(X)/G(X)=0
- Une erreur simple
  - $E(X)=x^i$; i identifie le bit en erreur
  - Condition à vérifier pour détecter cette erreur?

Types d'erreurs Détectées (cont.)

- Le receveur divise (T(X)+E(X)) par G(X)
  - T(X)/G(X)=0
- Une erreur simple
  - $E(X)=x^i$; i identifie le bit en erreur
  - Si G(X) contient 2 composantes ou plus, il ne va jamais diviser E(X)
    - Toutes les erreurs simples vont être détectées
- Une erreur double
  - $E(X)=x^i + x^j=x^j*(x^{i-j}+1)$; (i>j)
  - Si on suppose que G(X) n'est pas divisible par X,
  - C'est quoi la condition suffisante pour que toutes les erreurs doubles soient détectées?

Types d'erreurs Détectées (cont.)

- Une erreur double
  - $E(X)=x^i + x^j=x^j*(x^{i-j}+1)$; (i>j)
  - Si on suppose que G(X) n'est pas divisible par X, une condition suffisante pour que toutes les erreurs doubles soient détectées est que G(X) ne divise pas x^k+1 pour n'importe quelle k jusqu'à une valeur maximale de i-j (i.e., taille de la trame)
  - $G(X)=x^{15}+x^{14}+1$ ne divise pas x^k+1 pour n'importe quelle valeur inférieure à 32768

Types d'erreurs Détectées (cont.)

- Un nombre impair d'erreurs
  - E.g., $E(X)=x^4+x^3+1$
- Il n'y a pas de polynôme avec un nombre impair de composantes qui a x+1 comme facteur (dans la base binaire)
  - Preuve?
  - Que doit donc G(x) satisfaire?

Types d'erreurs Détectées (cont.)

- Preuve:
  - Supposons que E(X) a un nombre impair de composantes et est divisible par x+1
  - E(X)=(x+1)*Q(X)
  - E(1)=1
  - (1+1)*Q(1)=0
  - 1=0!!!!!!
  - Ce résultat est impossible puisque E(X) a un nombre impair de composantes
- En faisant X+1 un facteur de G(X), toutes les erreurs qui forment un nombre impair de bits inversés sont détectables

Types d'erreurs Détectées (cont.)

- Un code polynomial avec r bits de contrôle va détecter toutes les erreurs en rafales de longueur <=r
- Preuve?

Types d'erreurs Détectées (cont.)

- Preuve:
  - $E(X)=x^i*(x^{k-i}+\ldots.+1)$;
  - Donc si le degré de l'expression entre parenthèse est inférieur à celui de G(X), le reste ne peut jamais être égale à 0

Types d'erreurs Détectées (cont.)

- Si l'erreur en rafale est égale à r+1
- Le reste va être égale à 0 si et seulement si la rafale est égale à G(X)
  - Probabilité d'une telle éventualité (qu'une telle erreur passe inaperçue) est égale à $1/(2^{r-1})$
- Il peut être montré que la probabilité que des erreurs en rafales de longueur supérieure à r+1 passent inaperçues est petite $1/2^{r}$)

Types d'erreurs Détectées (cont.)

- CRC-16 = $x^{16}+x^{15}+x^2+1$
- Détecte toutes les erreurs simples, doubles, erreurs avec un nombre impair de bits inversés, toutes les erreurs en rafales de longueur <= 16, 99.997% d'erreurs en rafale de longueur égale à 17, 99.998% d'erreurs en rafales de longueur >=18

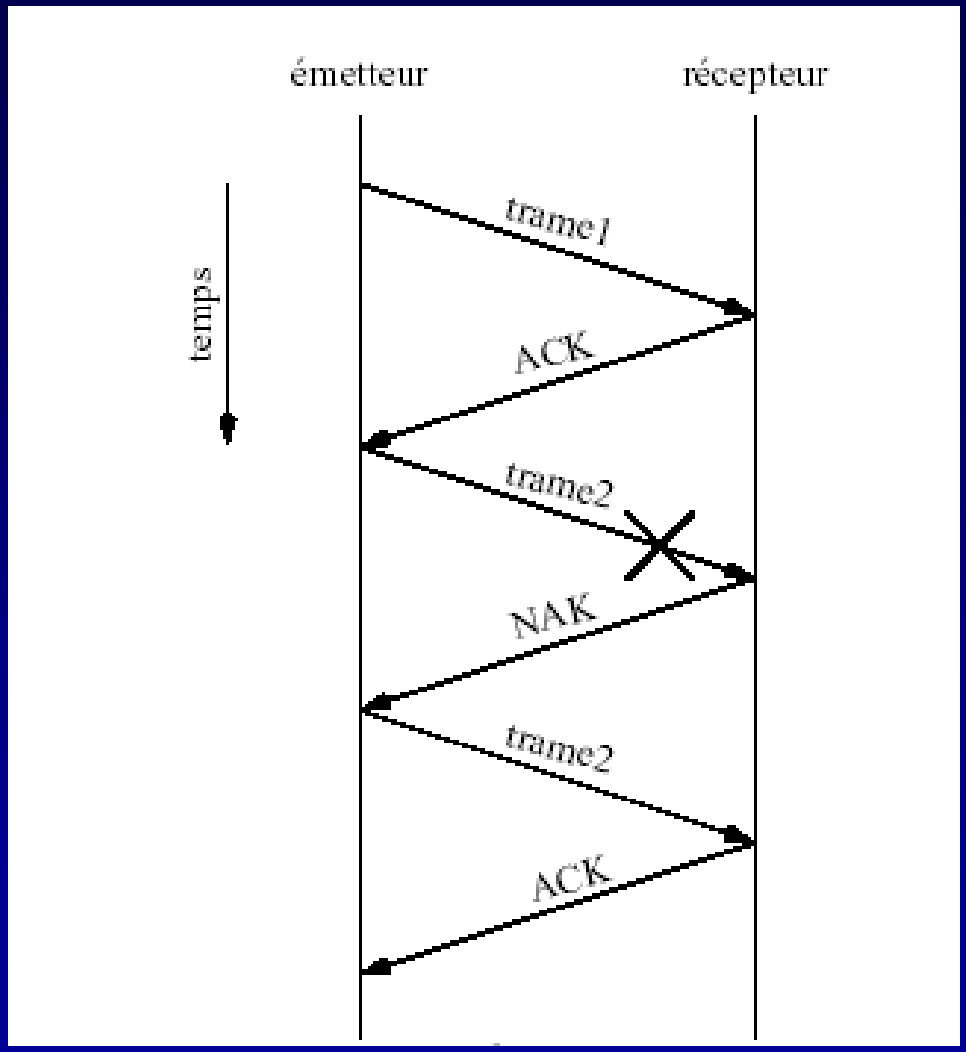
Contrôle de Flux

- Le contrôle du flux assure que l'émetteur n'envoie pas plus que le récepteur ne puisse recevoir.
- comment?

Protocoles ARQ (1)

- Envoyer et attendre (Stop and wait)
- Trame erronée?

Protocoles ARQ (2)

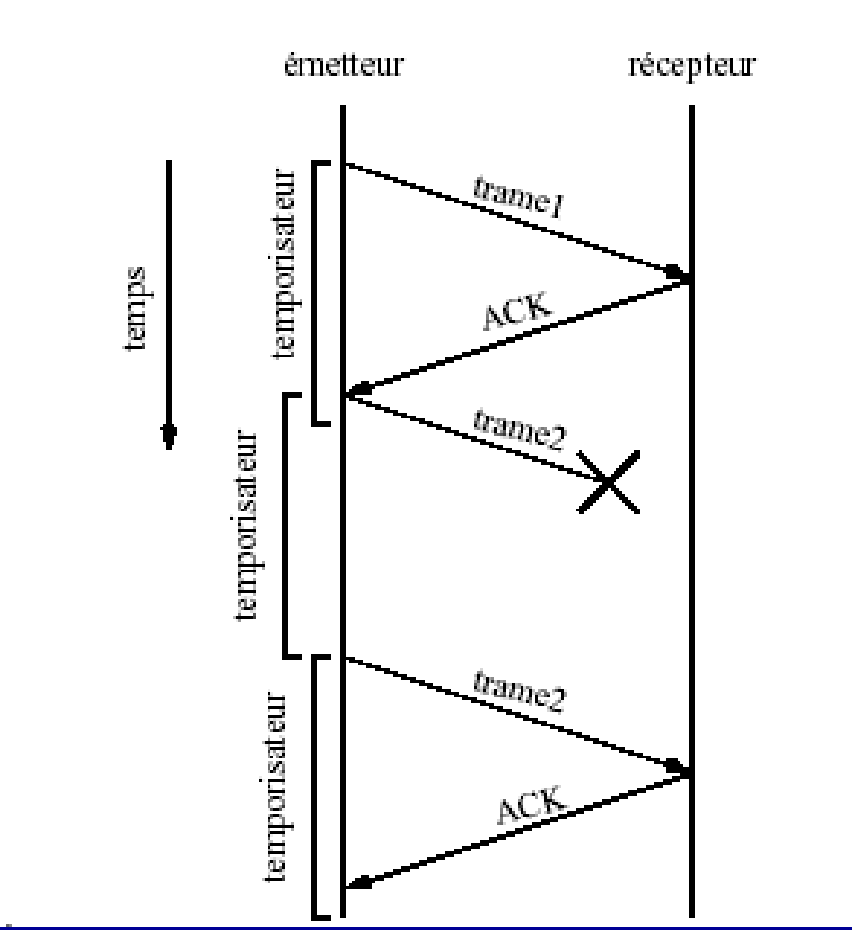


Trame erronée: acquittement positif et négatif

Protocoles ARQ (3)

- Trame perdue

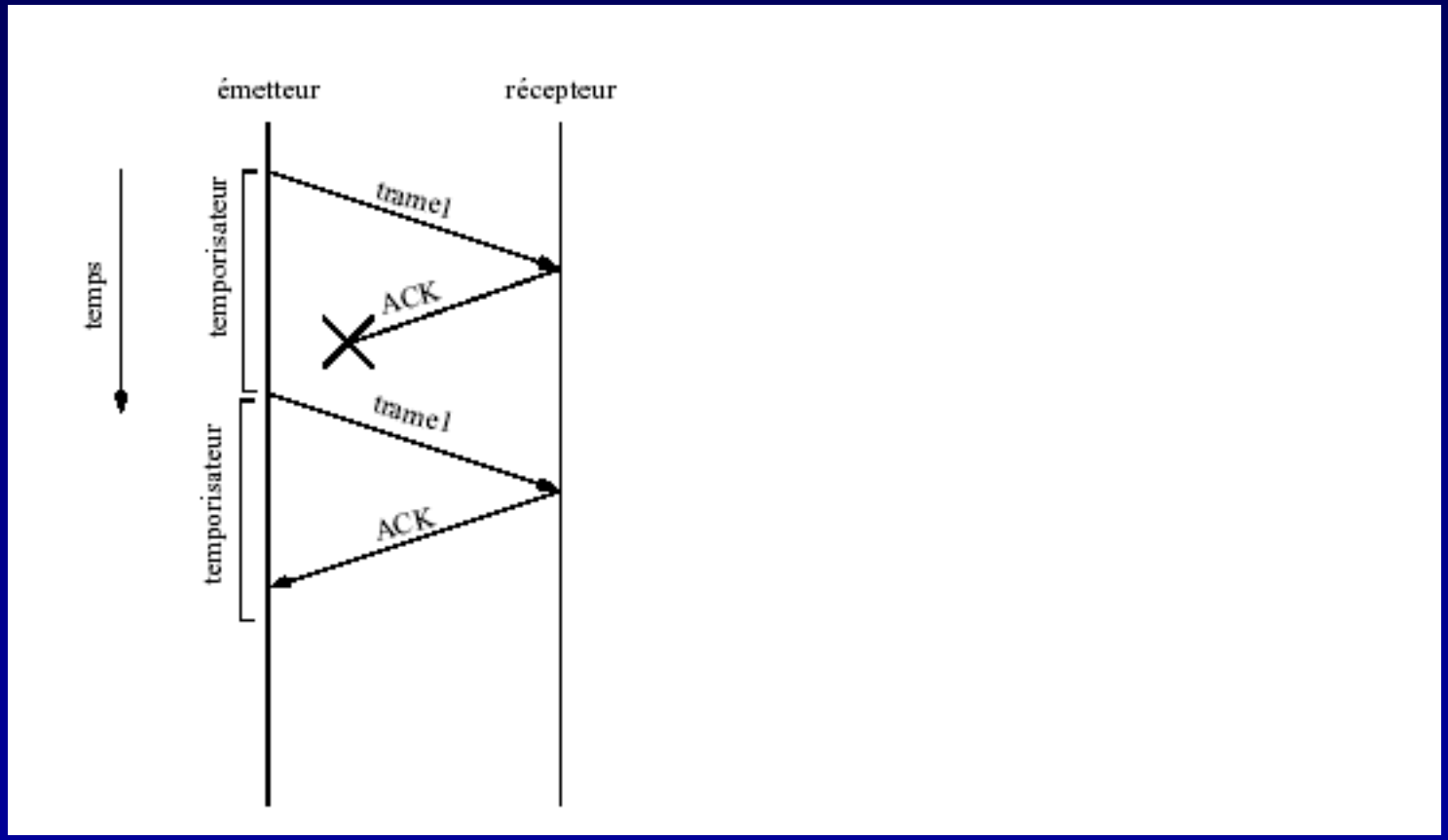
Protocoles ARQ (4)



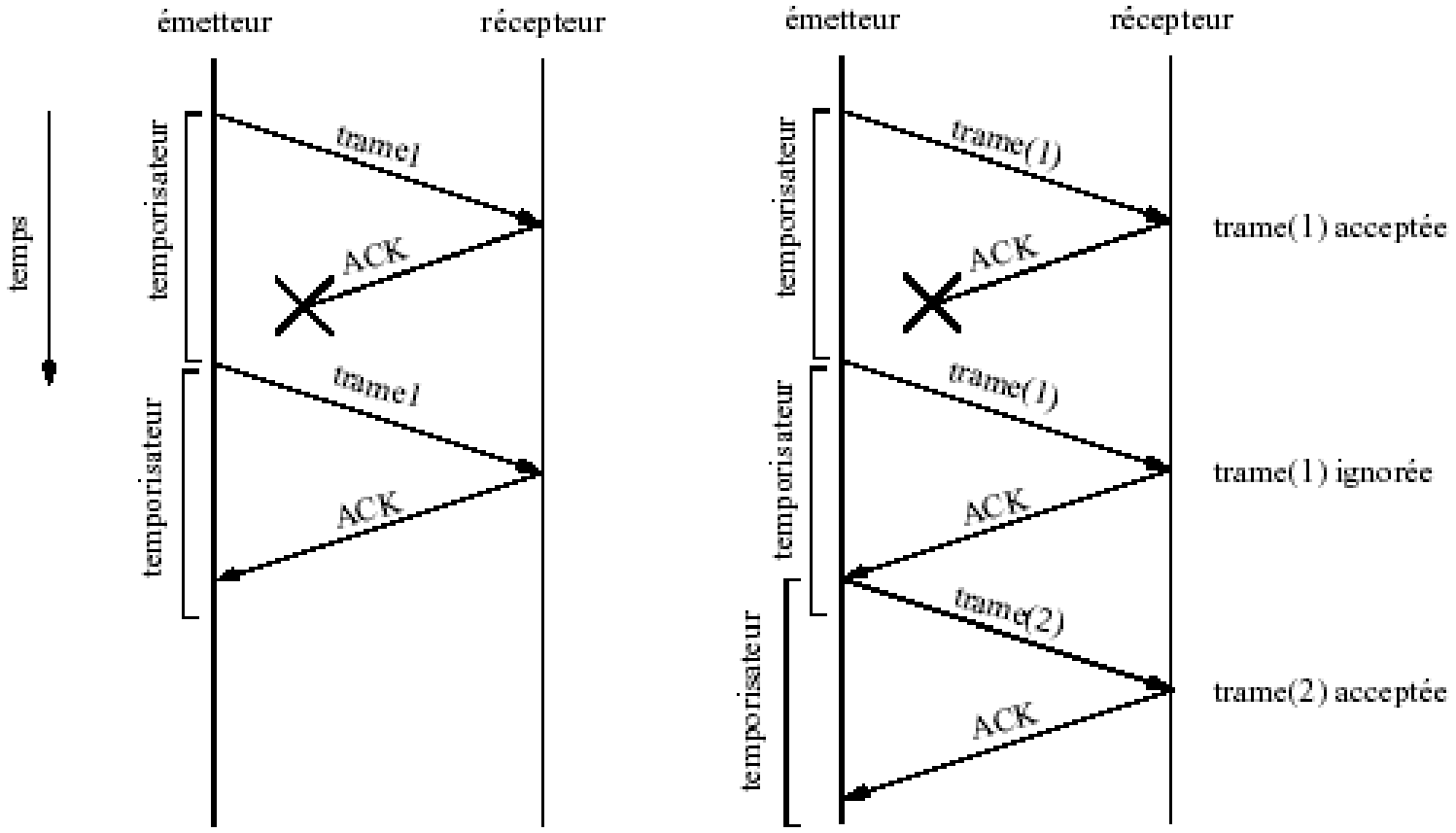
Trame perdue: temporisateur

Protocoles ARQ (5)

- Acquittement perdu, duplication



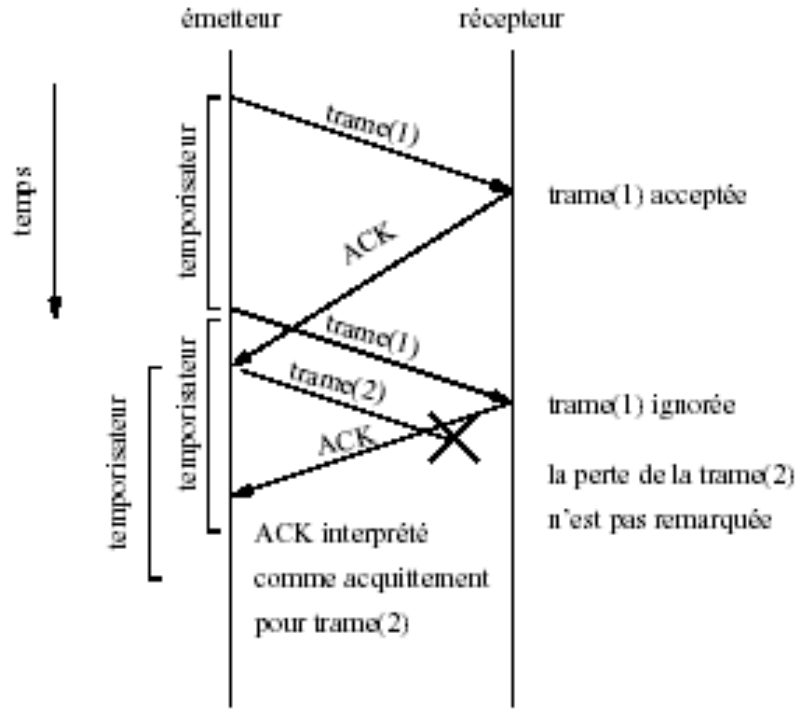
Protocoles ARQ (6)



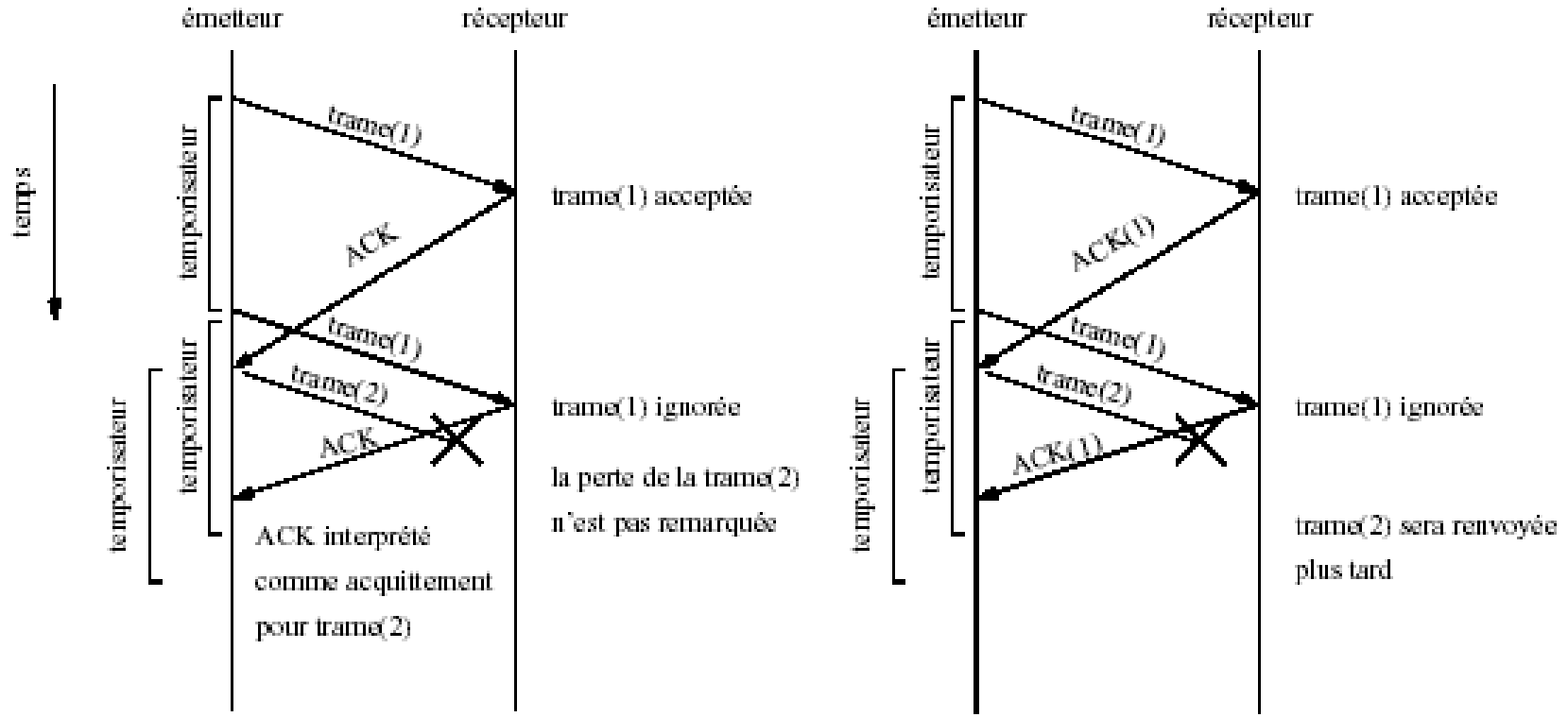
Acquittement perdu, duplication: numérotation des trames

Protocoles ARQ (7)

- Temporisateur expire trop tôt

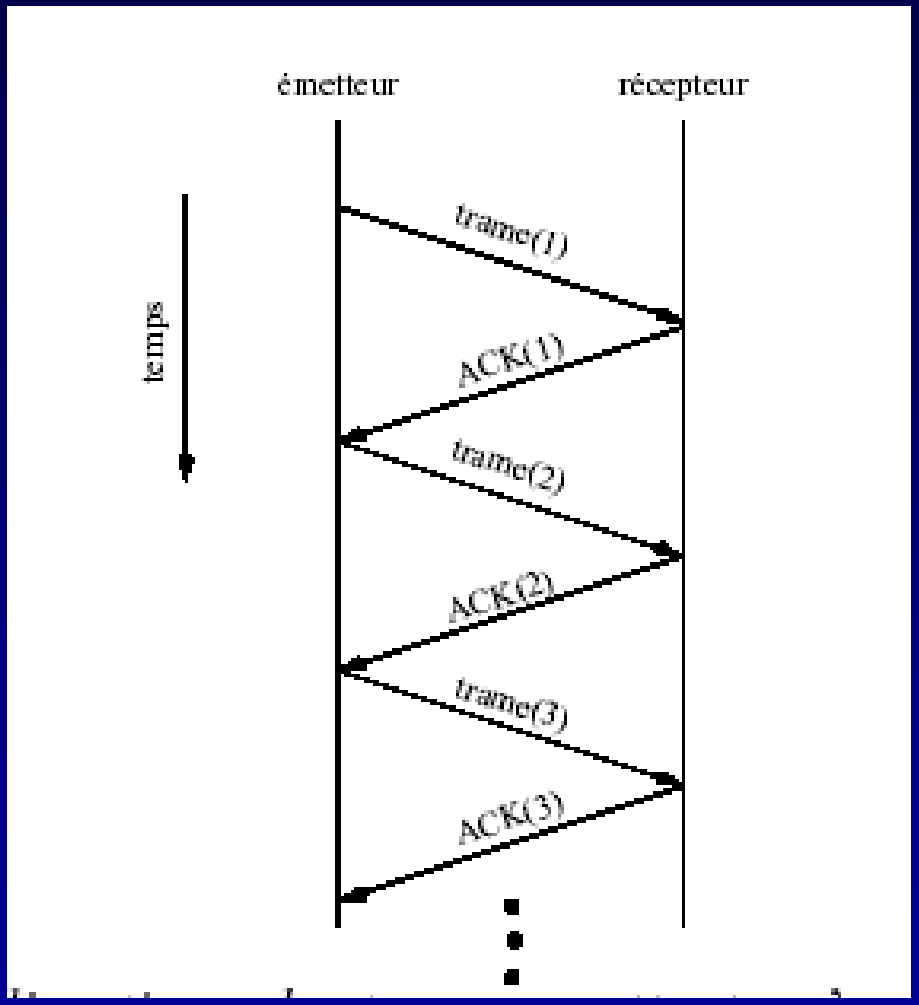


Protocoles ARQ (8)



Temporisateur expire trop tôt: numérotation des acquittements

Protocoles ARQ: envoyer et attendre



- Problème? Facteur/paramètre aggravant?

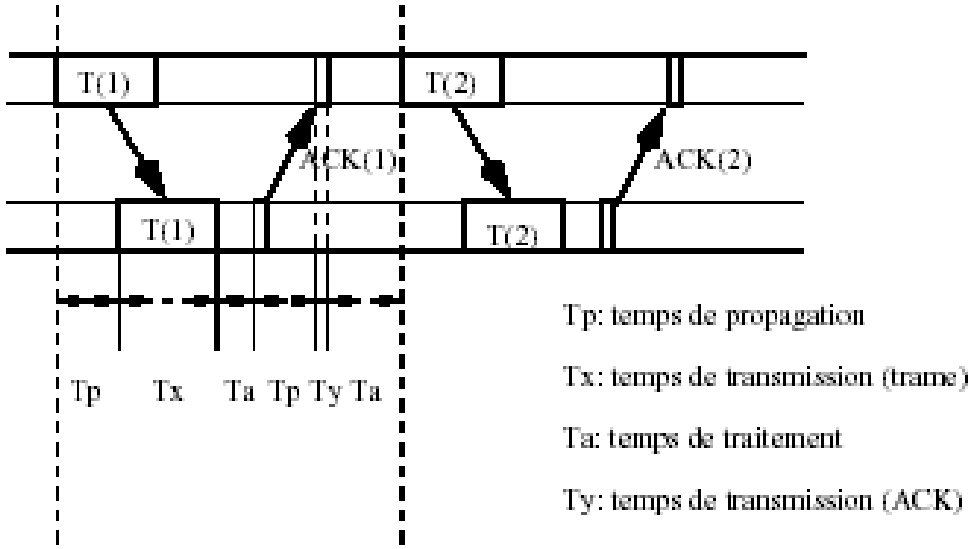
Protocoles ARQ: envoyer et attendre

- Pauvre utilisation du canal de transmission
- Paramètre aggravant: temps allez retour

Utilisation du canal par ’envoyer et attendre’

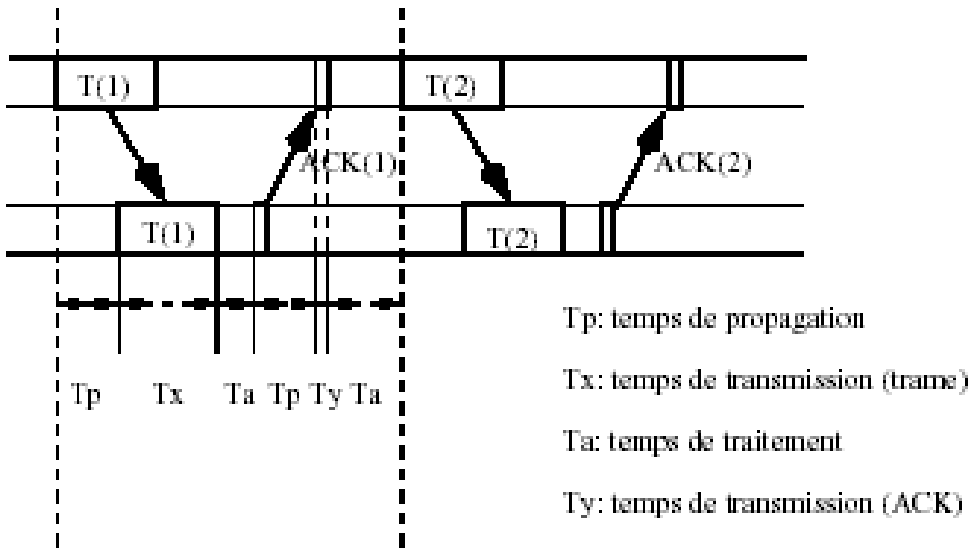
- **Exemple:** Ligne satellite à 56kbps et trames de 1000 bits. Le satellite est stationné à 33.000km sur la terre et la vitesse de propagation $3 * 10^8$ m/sec.
- **Utilisation:** Comment calculer l’utilisation U du canal en utilisant un protocole ’envoyer et attendre’ pour envoyer la trame et renvoyer l’acquittement?

Utilisation du canal par 'envoyer et attendre'



Comment la calculer?

Utilisation du canal par 'envoyer et attendre'



Utilisation: $U = \frac{Tx}{Tt}$ $Tt = Tx + Ty + 2Tp + 2Ta$

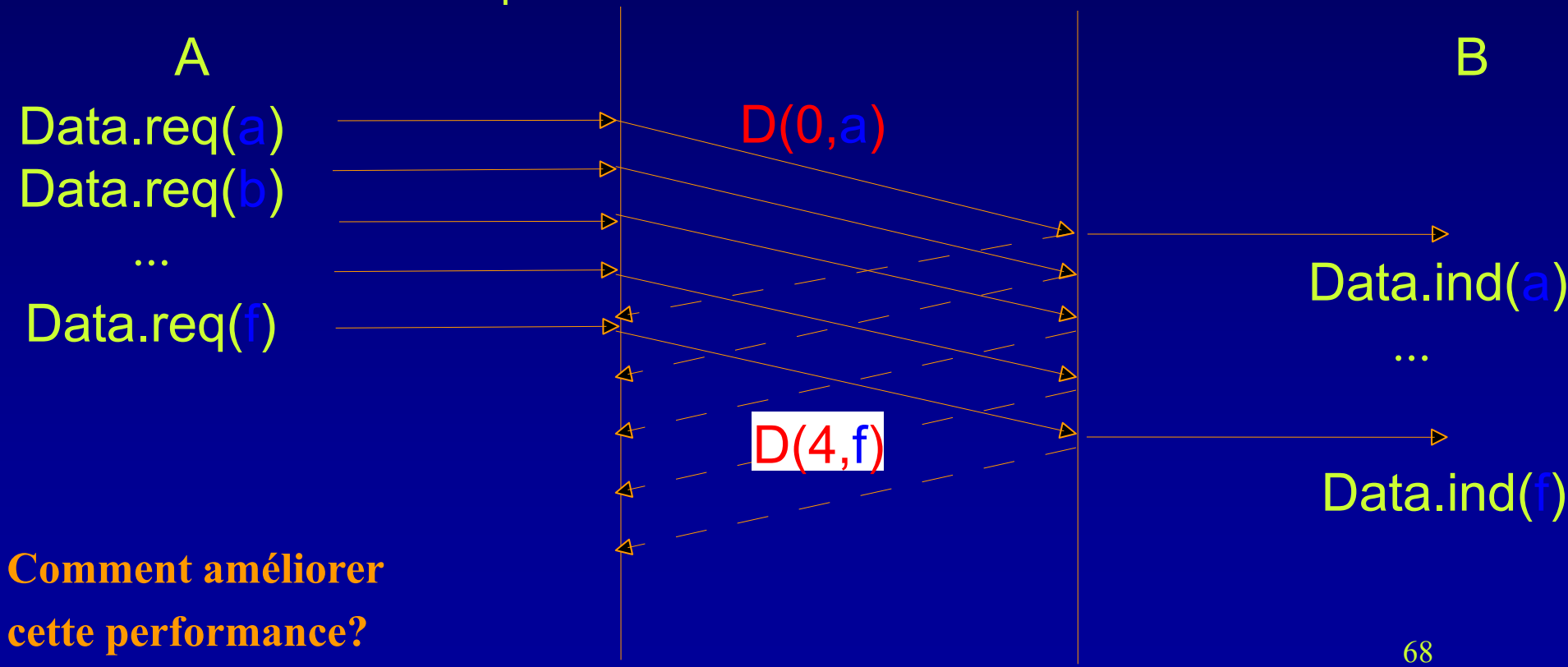
- **U=3.4%**
- **Comment améliorer cette performance?**

Protocoles ARQ (Automatic Repeat Request): envoyer et attendre

- Que faire? Comment améliorer la performance?

Amélioration des performances de 'envoyer et attendre'

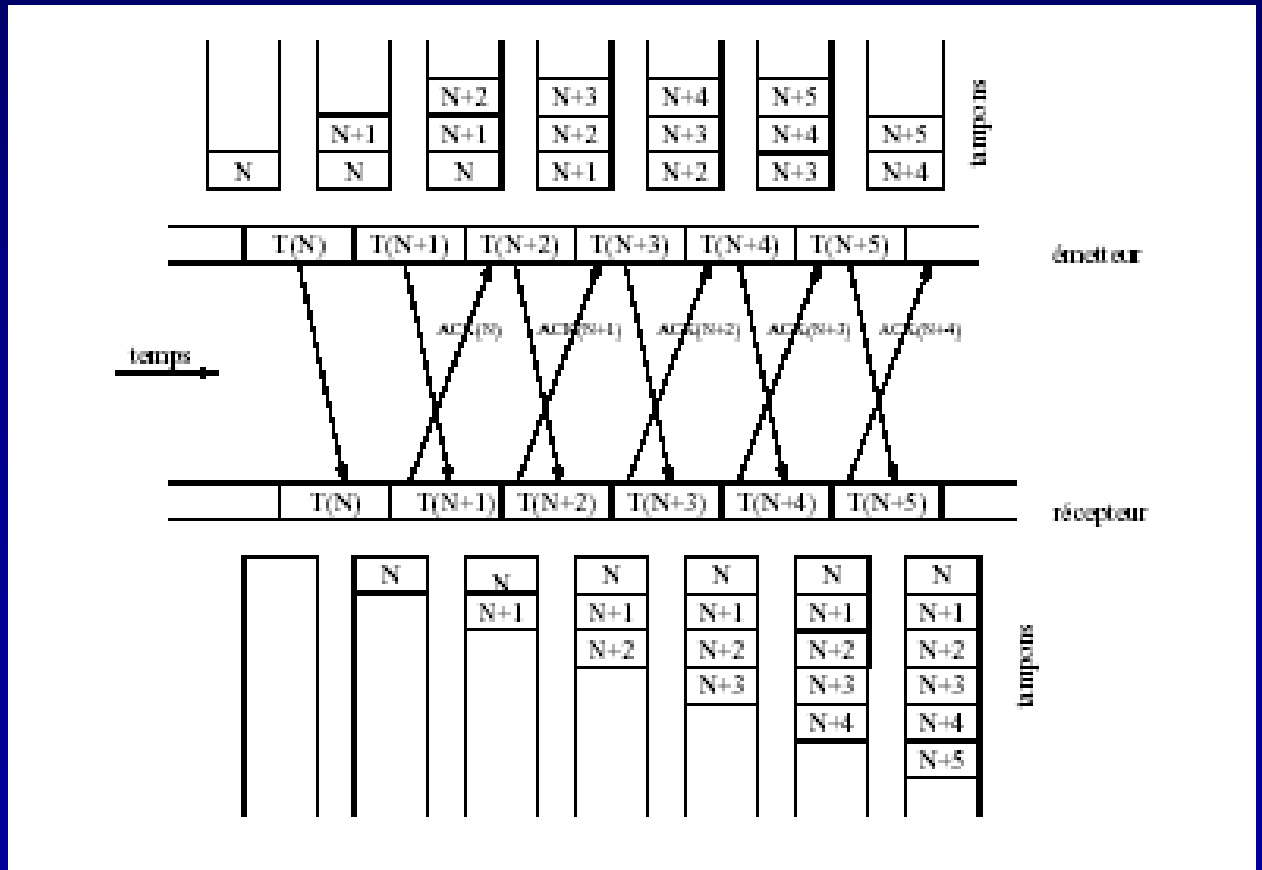
- Technique du pipeline
- Principe
  - permettre à l'émetteur d'envoyer plus d'une trame en attendant l'acquit du receveur



Comment améliorer cette performance?

Amélioration des performances de 'envoyer et attendre' (Cont.)

- Pour utiliser encore davantage la capacité du canal, on passe à un mode bidirectionnel ou les acquittements sont ajoutés à des trames envoyées dans l'autre direction (*piggybacking*): besoin de tampons pour les trames non-acquittées.



Comment améliorer cette performance?

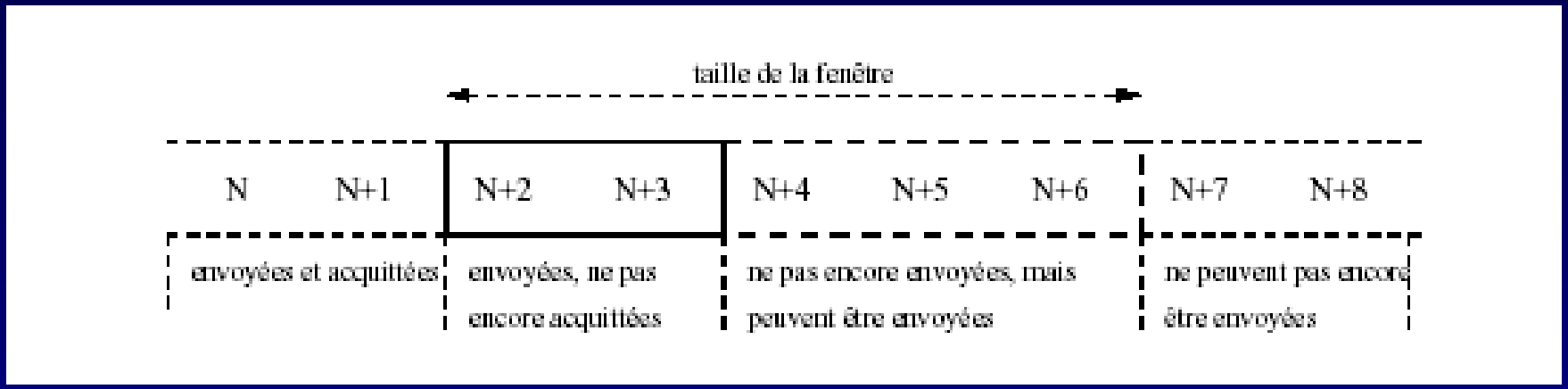
Amélioration des performances de ’envoyer et attendre’ (Cont.)

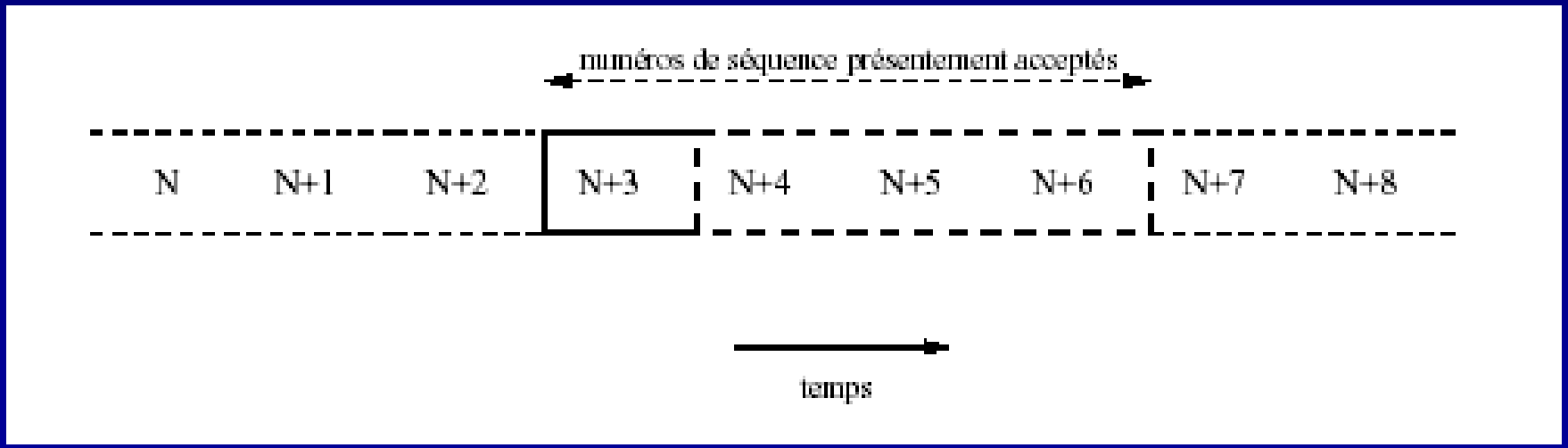
- Acquittement cumulative
  - Acquitter plusieurs trames avec un seul acquittement

Fenêtre d'anticipation/glissante/coulissante

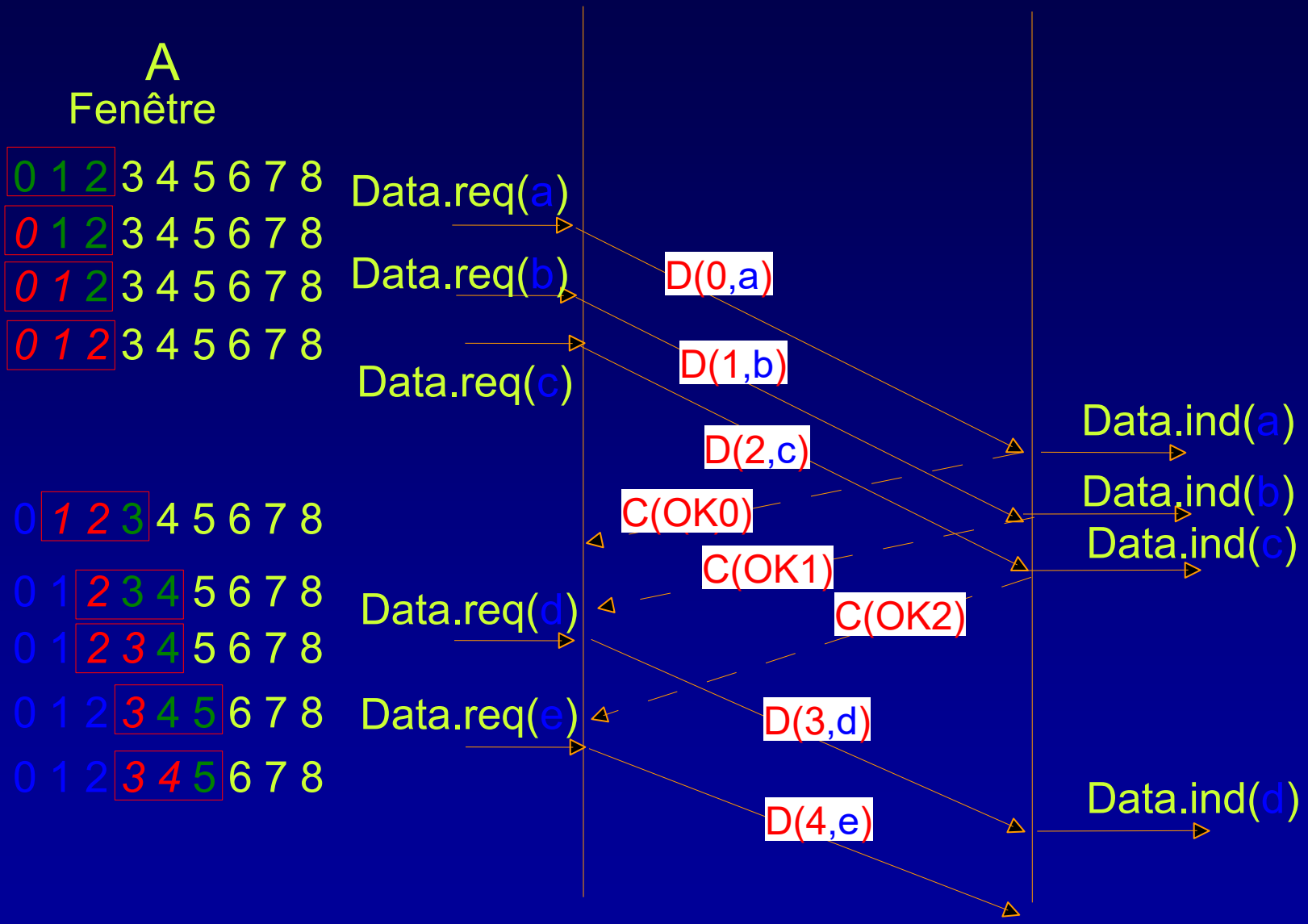
- Principe
  - À tout moment, l'émetteur maintient une liste de numéros de séquence consécutifs qu'il peut envoyer
    - fenêtre d'émission (sending_window)
  - À tout moment le receveur maintient une liste de numéros de séquence consécutifs qu'il peut recevoir
    - fenêtre de réception (receiving_window)
  - L'émetteur et le receveur doivent choisir leurs fenêtres respectives de façon cohérente
    - généralement sending_window = receiving_window
    - généralement cette taille est fixée pour un protocole donné ou parfois négociée lors de l'établissement de la connexion

Fenêtre de l' émetteur et le récepteur





Fenêtre glissante : exemple 1



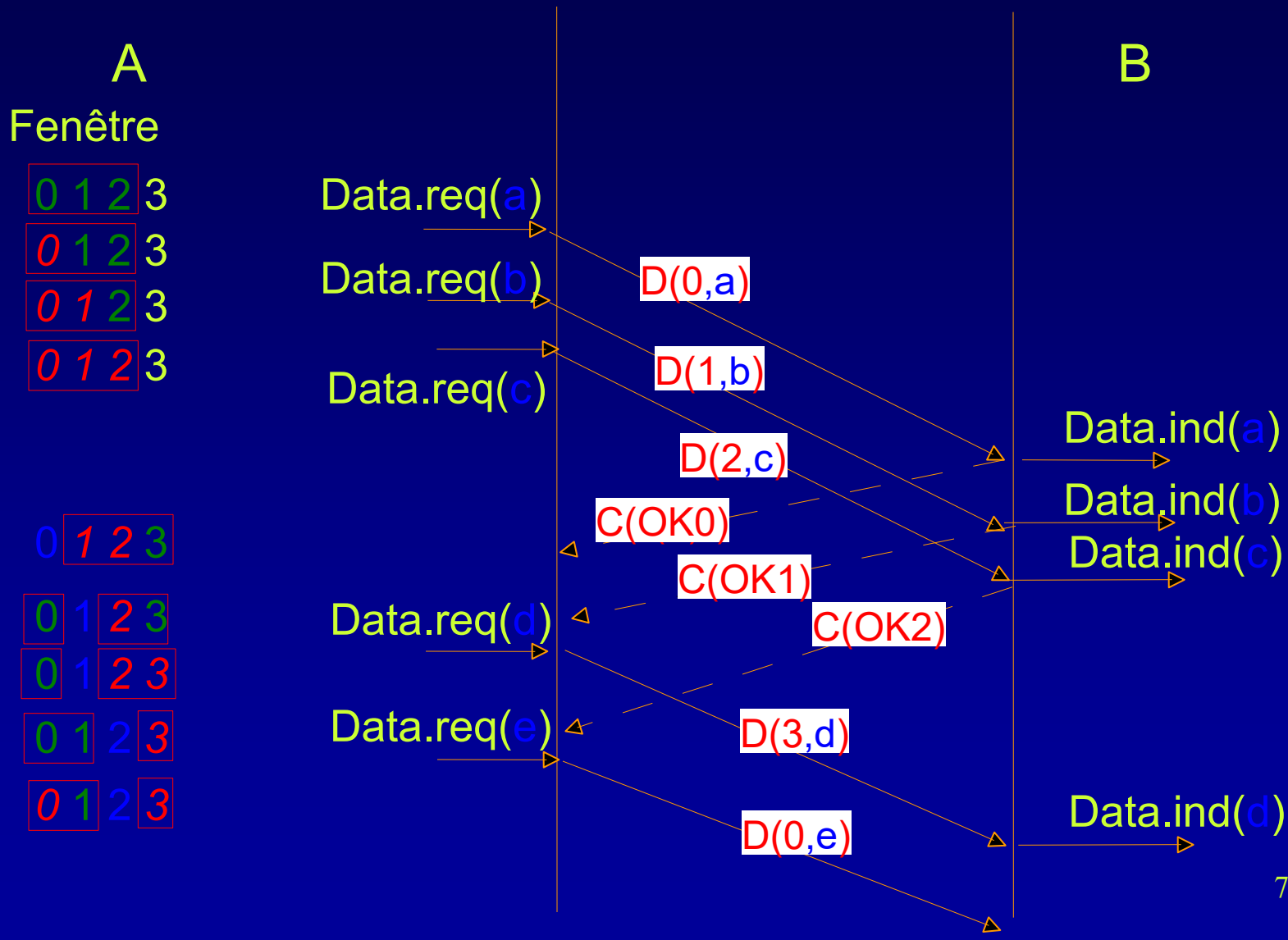
Fenêtre glissante : numérotation

- Problème
  - Pour placer le numéro de séquence dans la trame, il faut l'encoder sur N bits
    - 2^N numéros de séquence distincts
    - 1 000 000 de trames ou plus
    - Surcharge
- Solution?

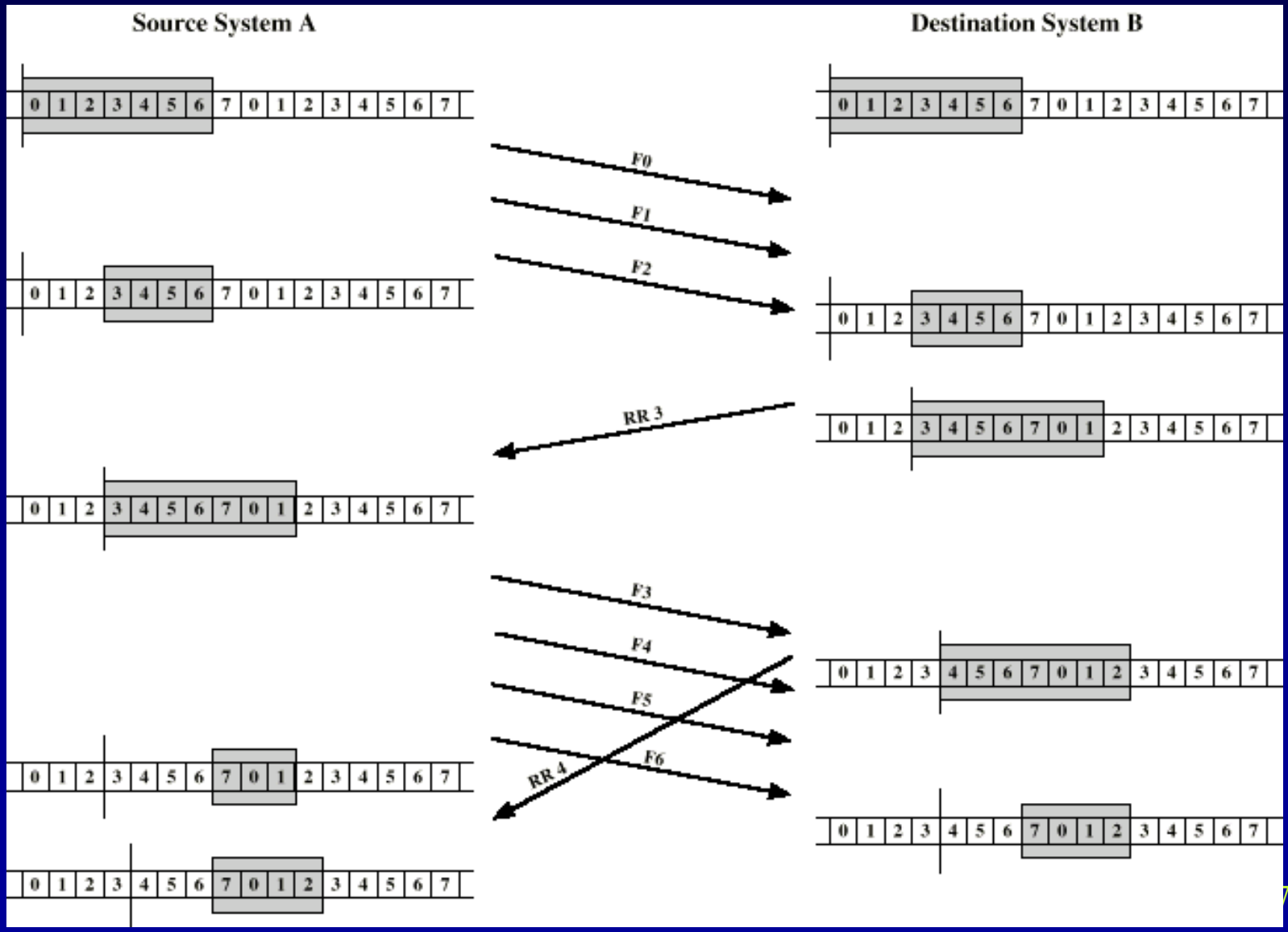
Fenêtre glissante : numérotation

- Solution
  - Placer dans chaque trame (données et contrôle) le numéro de séquence modulo 2^N
  - on réutilisera donc le même numéro de séquence pour des trames distinctes
    - problèmes !
  - Fenêtre d'émission
    - Liste de numéros de séquence consécutifs (modulo 2N) que l'émetteur peut envoyer

Fenêtre glissante : exemple 2



Fenêtre glissante : exemple 3



Fenêtre glissante et transfert fiable

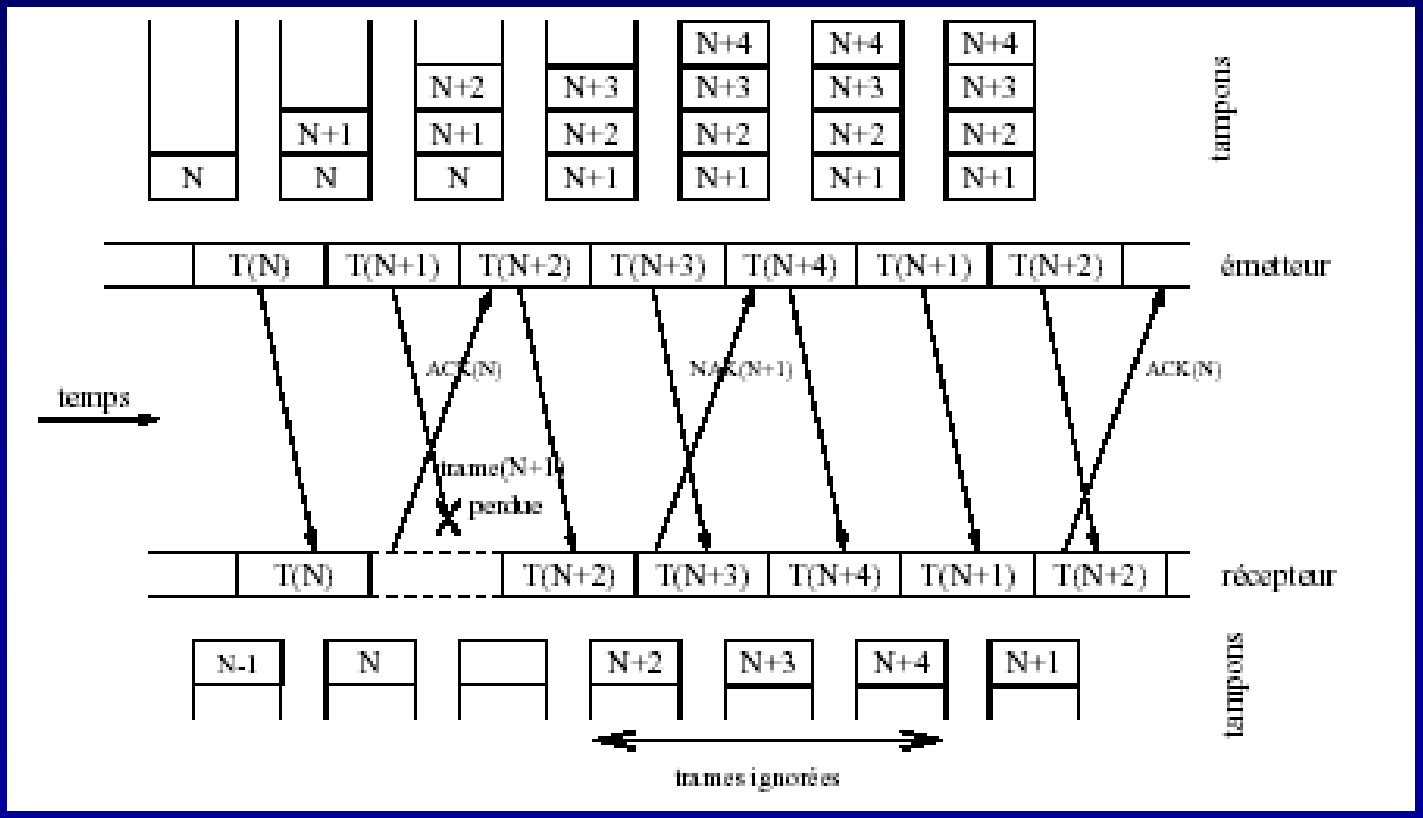
- Problème
  - Comment fournir un transfert fiable tout en utilisant une fenêtre glissante ?
    - Sémantique des trames de contrôle
    - Fonctionnement de l'émetteur et du receveur
- Solutions (description de protocoles qui peuvent implémenter un transfert fiable en utilisant tout ce qu'on a couvert)?

Fenêtre glissante et transfert fiable

- Solutions
  - Go-Back-N
    - simple à implémenter (surtout au receveur)
    - les performances chutent si les erreurs sont fréquentes
  - Selective Repeat
    - plus complexe à implémenter (pour émetteur et receveur)
    - meilleures performances que go-back-n lorsque le taux d'erreurs de la couche physique est élevé

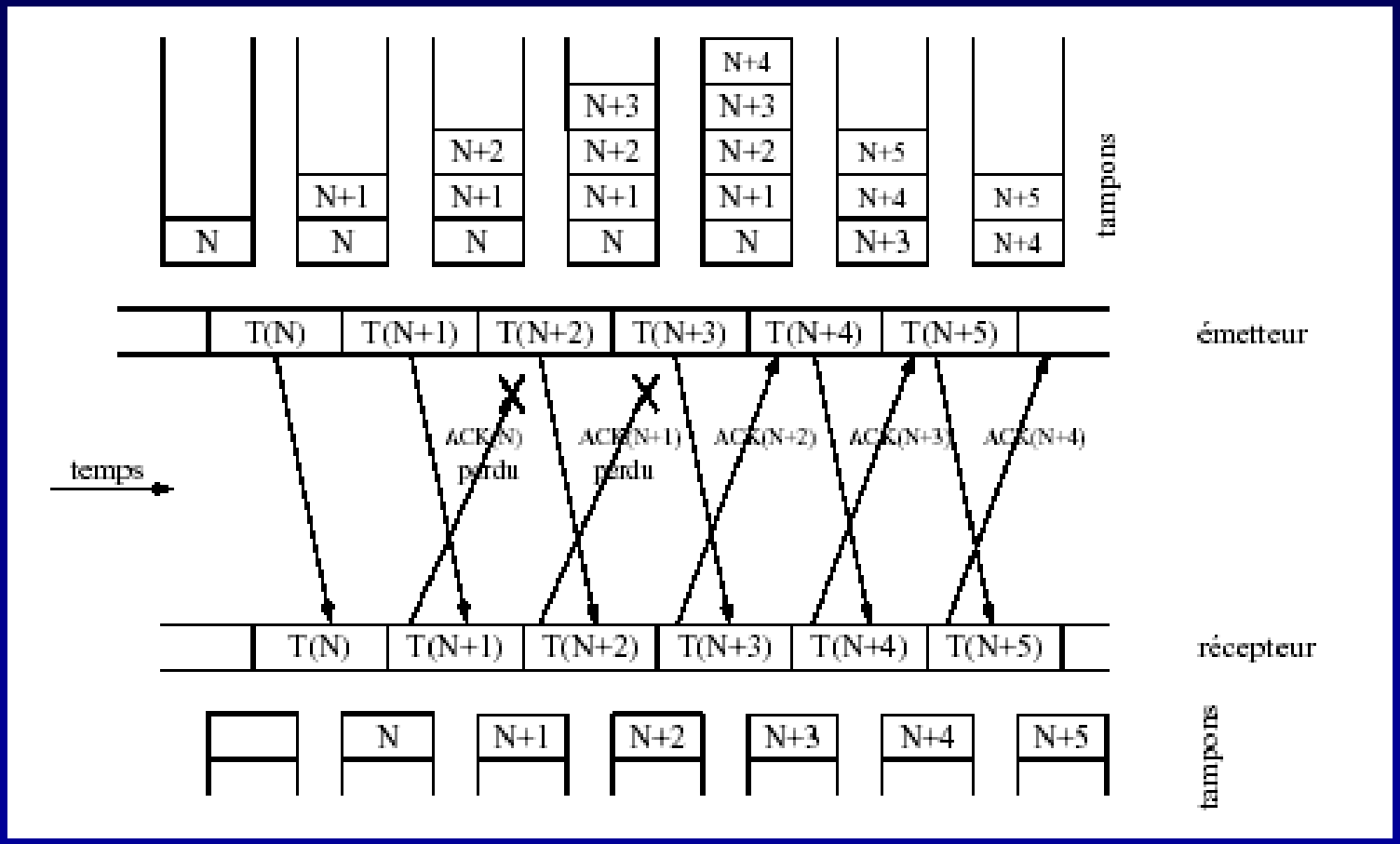
Contrôle d'erreur: Go-back-N

- Chaque acquittement est cumulatif: `ACK(N)` acquitte toutes les trames jusqu'au numéro `N`
- L'acquittement négatif `NAK(N+1)` demande la retransmission à partir de la trame `T(N+1)`.
- Chaque `NAK` est associé avec un temporisateur pour permettre sa retransmission en cas de perte.



Contrôle d'erreur: Go-back-N (Cont.)

- Si des acquittements sont perdus, la trame `ACK` suivante acquitte aussi les trames précédentes



Go-back-N: Taille Maximale de la fenêtre

- Suppose 3 bits utilisés pour la numérotation
  - 0, ….7
- Taille maximale: 8?

Go-back-N: Taille Maximale de la fenêtre (cont.)

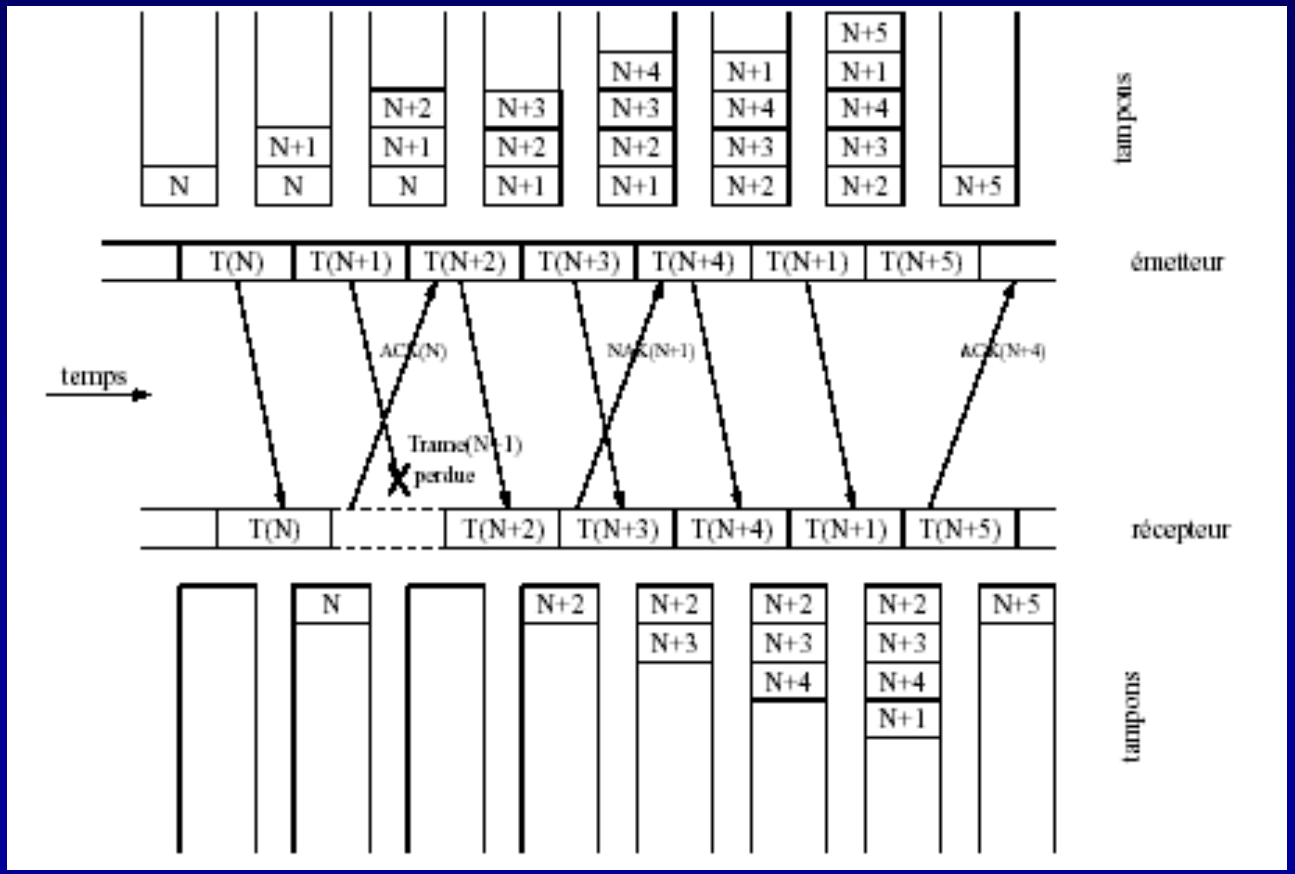
- Suppose la source envoie les trames 0 …7
- La destination envoie une confirmation (piggybacking) que les 8 trames sont reçues correctement
  - La confirmation est reçue par la source
- Suppose la source envoie un autre ensemble de 8 trames: 0 …7
  - Toutes les 8 trames sont perdues
- La destination envoie une confirmation (piggybacking) que les 8 trames sont reçues correctement
  - La source n'a aucune connaissance est ce que les 8 dernières trames ont été reçues
-

Go-back-N: Taille Maximale de la fenêtre (cont.)

- Suppose la source envoie les trames 0 …7
- La destination envoie une confirmation que les 8 trames sont reçues correctement
  - La confirmation est perdue
- La source renvoie les mêmes trames 0 … 7 après l'expiration du temporisateur
- La destination envoie une confirmation que les 8 trames sont reçues correctement
  - La destination n'a aucune connaissance que les 8 dernières trames sont une duplication!
- **Taille est $2^n - 1$**

Contrôle d'erreur: Rejet sélectif

- Chaque acquittement est cumulatif: `ACK(N)` acquitte toutes les trames jusqu'au numéro `N`
  - Chaque `NAK` est associé avec un temporisateur pour permettre sa retransmission en cas de perte.



Rejet sélectif : Taille Maximale de la fenêtre

- Suppose 3 bits utilisés pour la numérotation
  - 0, ….7
- Taille maximale: 7?