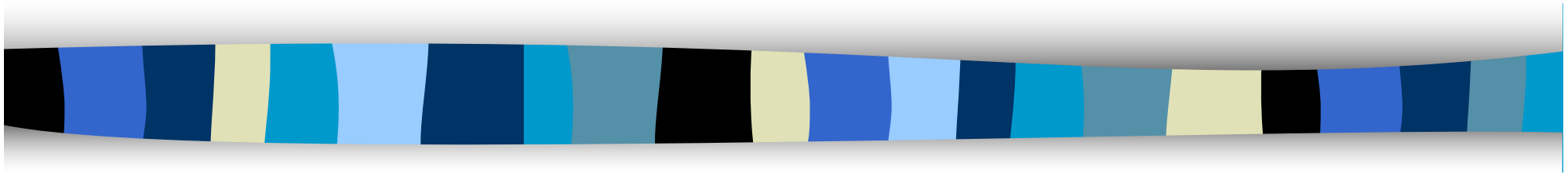
Liaison logique



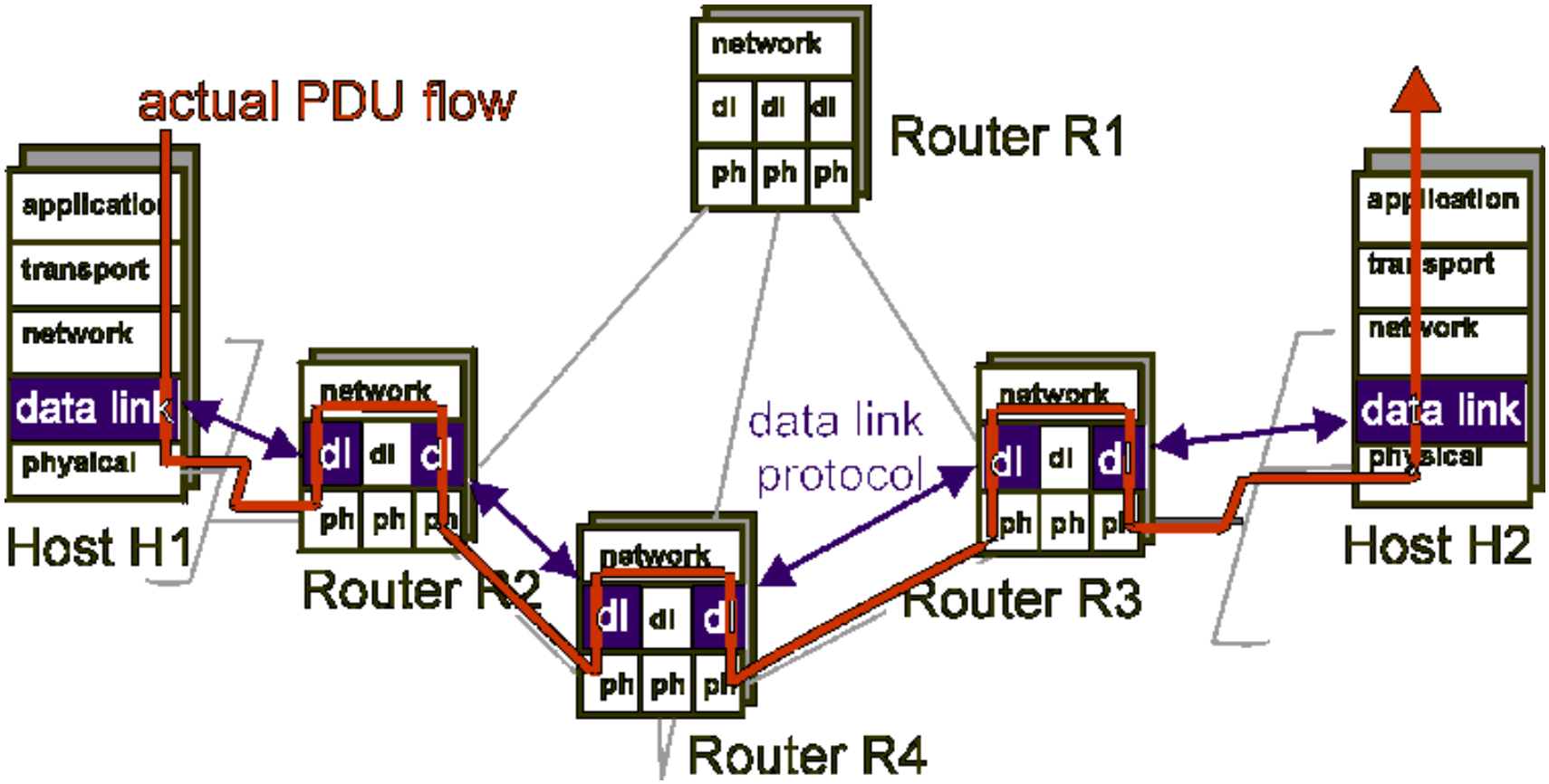
C. Pham
Université de Pau et des Pays de l'Adour
Département Informatique
http://www.univ-pau.fr/~cpham
Congduc.Pham@univ-pau.fr

Copyright

Plan

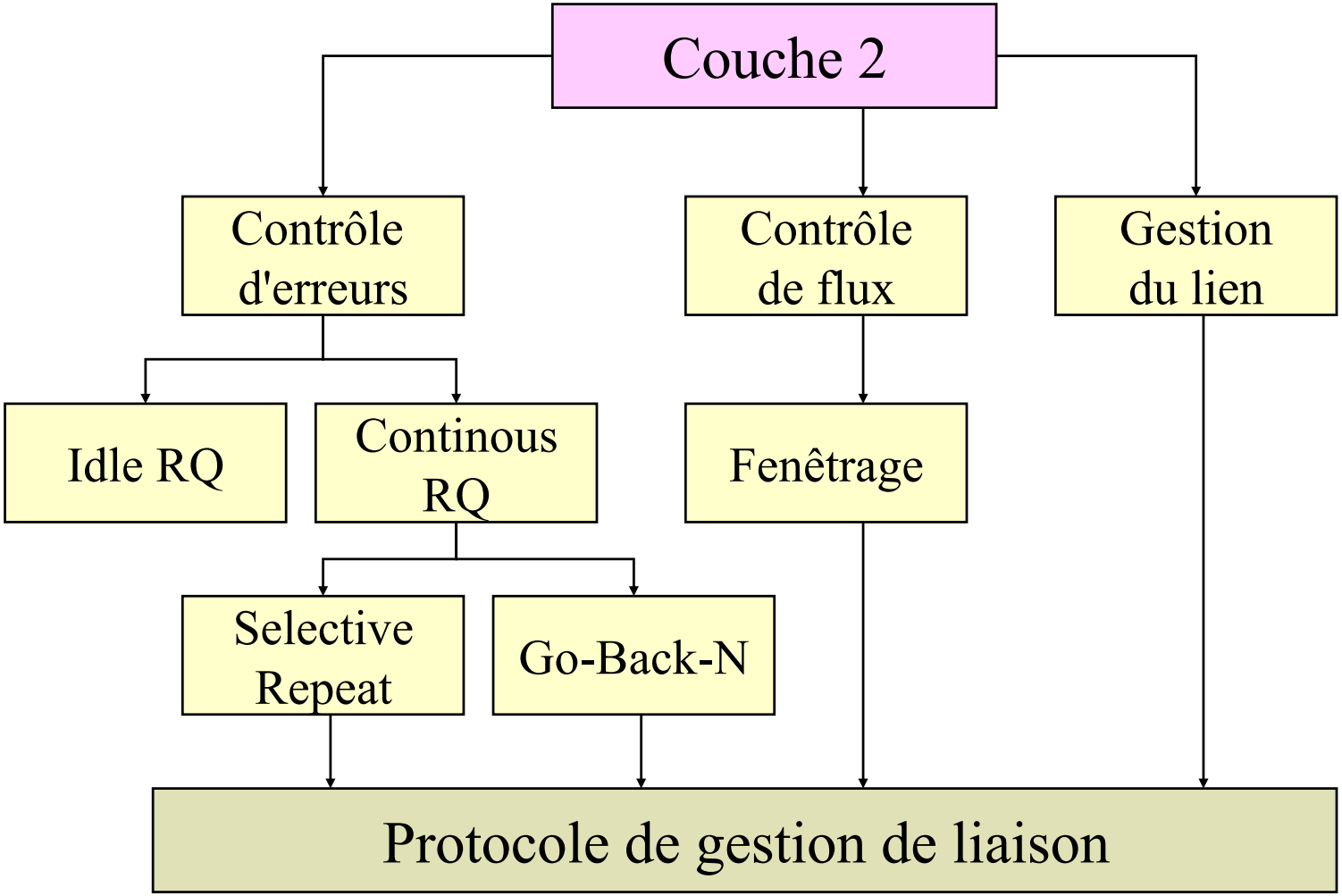
- **Introduction**
- **Trames et protocoles de gestion de liaison**
- **Contrôle et détection d'erreurs**
  - parité, CRC
  - code de Hamming
- **Mécanismes de retransmission ARQ**
  - Idle RQ
  - Continous RQ
- **Mécanismes de contrôle de flux**
  - Sliding windows
- **BSC, HDLC**
- **Sous-couche MAC**

Où se situe la couche liaison?



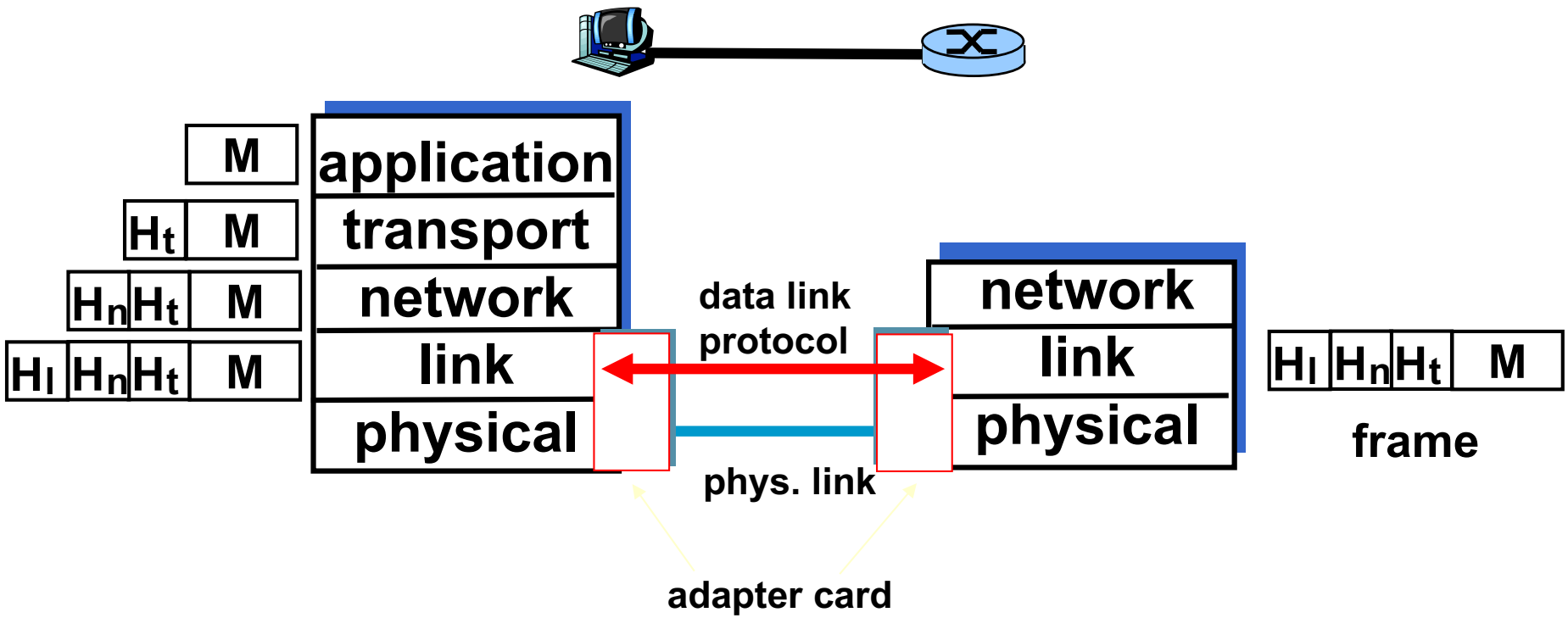
Les fonctions de la couche liaison

- La couche 2 fournit les composantes suivantes:

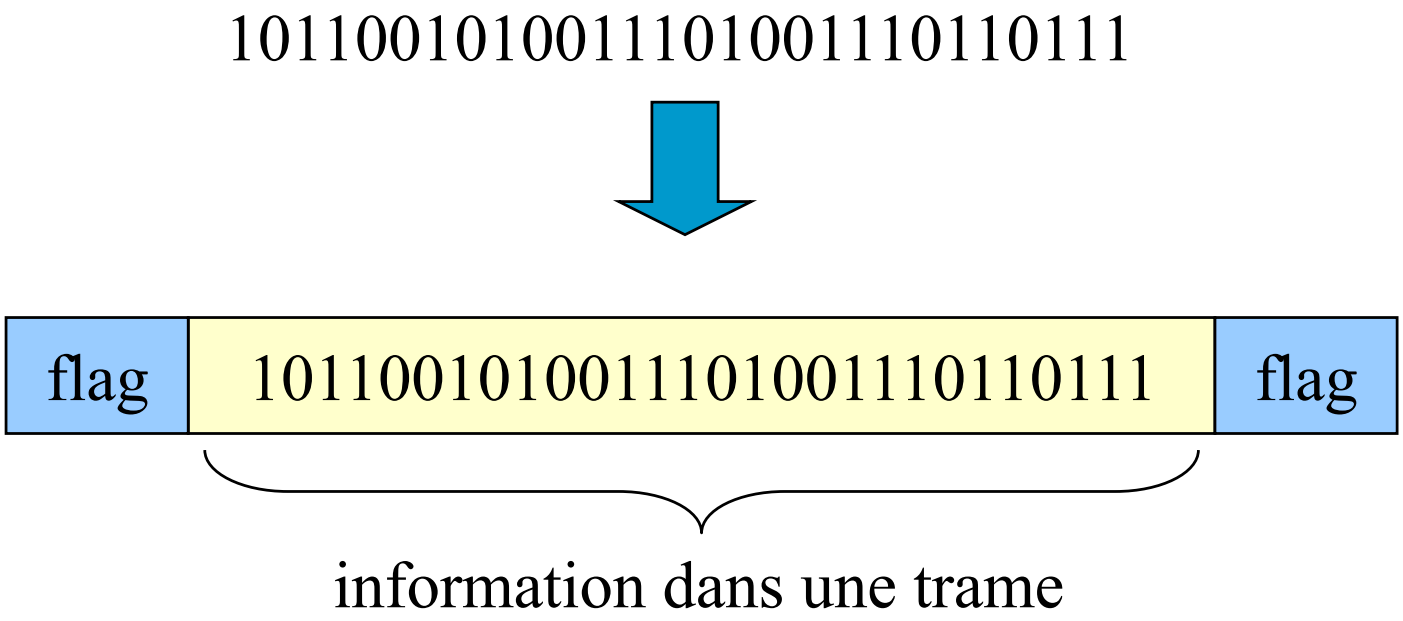


Implémentation

- **Implémentée le plus souvent dans les "cartes"**
  - e.g., carte Ethernet, carte modem, carte WiFi…
  - contient: RAM, DSP, interfaces bus, et interfaces lien



La notion de trame



- **Le tramage permet**
  - de découper le flot d'information
  - de partager le support par plusieurs applications/machines
  - de rajouter des mécanismes de contrôle

➡ **Protocoles de gestion de liaison**

Protocoles de gestion de liaison

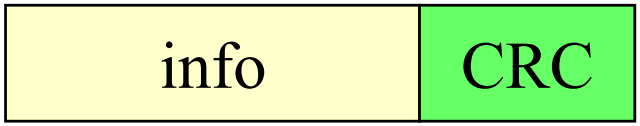
- **Un protocole de gestion de liaison introduit**
  - un contrôle d'erreur: détection et correction d'erreur
  - des mécanismes de reprise sur erreurs (Idle RQ, Go-Back-N…)

- **Orienté caractère**
  - BSC: Binary Synchronous Control (Idle RQ)

- **Orienté bits**
  - HDLC: High-level Data Link Control (Continous RQ)

Contrôle et détection d'erreurs

- **But: ajouter des bits de contrôle pour détecter et éventuellement corriger les erreurs**
- **Les codes systématiques**
  - Les bits de contrôle sont concaténés aux bits d'information
  - Ex: bits de parité, CRC



- **Les codes non-systématique**
  - Les bits de contrôle peuvent être mélangés à l'information
  - Ex: Codage de Hamming
- **L'addition se fait modulo-2**

+	0	1
0	0	1
1	1	0

Exemple simple, les bits de parité

- **Parité paire et Parité impaire**
  - un bit à 0 ou 1 est rajouter pour que la somme du nombre de 1 soit paire ou impaire
  - détection d'un nombre impair d'erreurs
- **Parité transversale+longitudinales**
  - Block Sum Check
  - row parity+column parity
  - permet de détecter plus d'erreurs
- **Le codage de Hamming**
  - code détecteur et correcteur d'une erreur
  - utilise des bits de parité à des positions $j=2^i$, i=0..n
  - code non-systématique

Mot de code et distance de Hamming

- Un mot de code c'est l'information + les bits de contrôle
- Exemple avec la parité paire: si S envoie 01011100 et R reçoit 01011110 en parité paire, ce n'est pas un mot de code valide.
- La distance de Hammings est le nombre de bits minimum qui diffèrent d'un mot de code valide d'un autre.

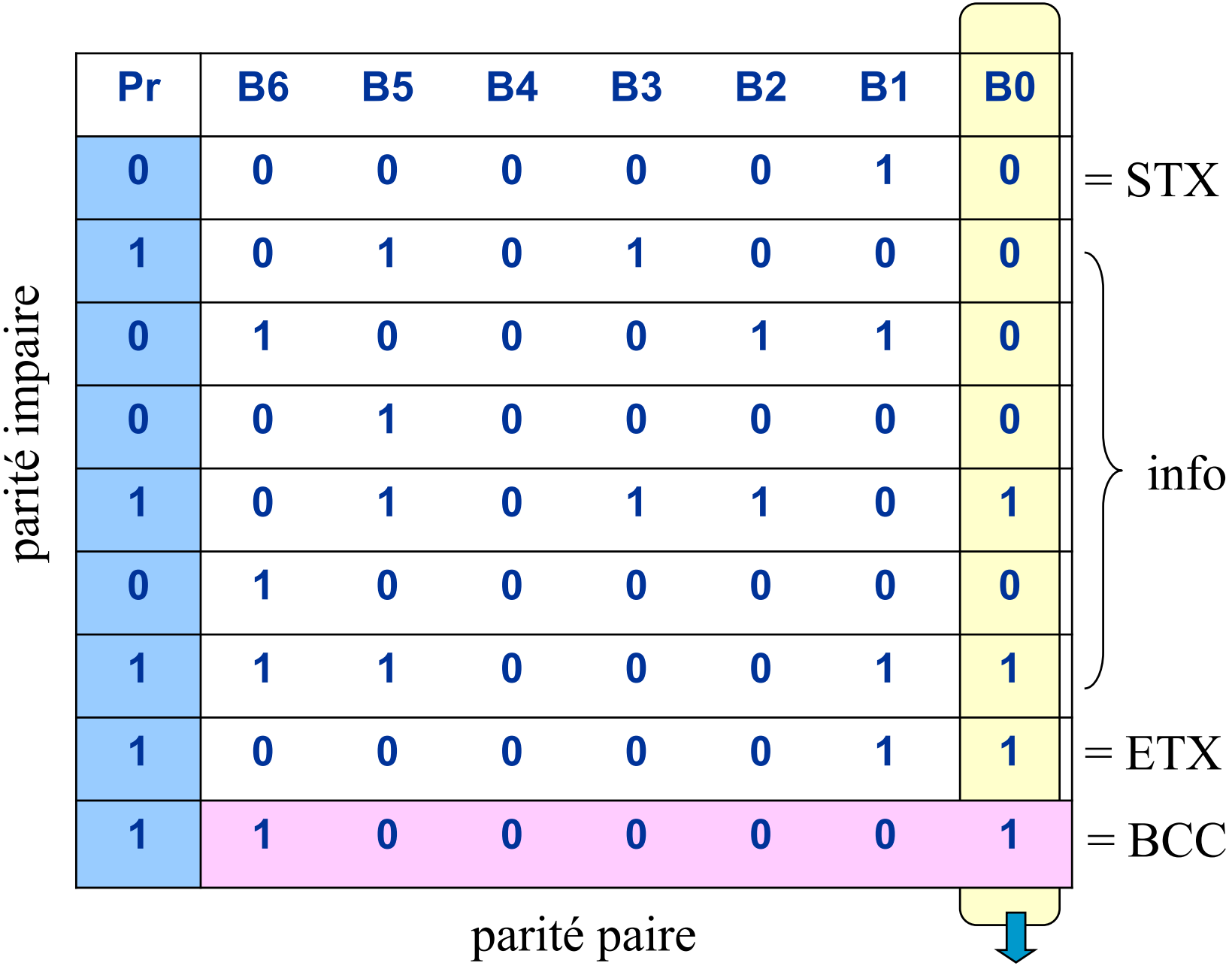
- Q: Quelle est la distance de Hammings d'un code de parité?

La distance de Hamming

- **Théorème:**
  - Pour détecter *d* erreurs, il faut que la distance de Hammings entre 2 mots de code valides soit de *d*+1
  - Pour corriger *d* erreurs, il faut que la distance de Hammings entre 2 mots de code valides soit de 2*d*+1

- **Comment expliquez vous intuitivement ces théorèmes?**
  - construisez par exemple 4 mots de code sur 10 bits dont la distance de Hammings est 5.

Block sum check



Pr	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	1	0
1	0	1	0	1	0	0	0
0	1	0	0	0	1	1	0
0	0	1	0	0	0	0	0
1	0	1	0	1	1	0	1
0	1	0	0	0	0	0	0
1	1	1	0	0	0	1	1
1	0	0	0	0	0	1	1
1	1	0	0	0	0	0	1

Les codes linéaires (1)

- **Si les bits de contrôle sont une combinaison linéaire des bits d'information, alors le code est linéaire**
  - ex: le code de Hamming est un code linéaire
  - L'addition de 2 mots de code donne un mot de code
- **Notation C(n,k)**
  - n est la longueur du mot de code
  - k est le nombre de bits d'information
- **Si $u=(u_0, u_1, u_2, u_3, u_4)$ le vecteur que l'on transmet et que le code est systématique de la forme MC on a:**

m_0	m_1	m_2	c_0	c_1
u_0	u_1	u_2	u_3	u_4

- **Sous forme matricielle ➡ u=m.M où M est la matrice génératrice du code**

Les codes linéaires (2)

- **Soit M=($I_{k,k}$, $P_{k,n-k}$), ex:** $M=\begin{pmatrix}100P_{00}P_{01}\\010P_{10}P_{11}\\001P_{20}P_{21}\end{pmatrix}$

- **Pour l'encodage on fait u=m.M**

- **Pour le décodage, on fait le calcul du syndrome**
  - Le récepteur reçoit X tel que $H.X^t=0$, H est la matrice de contrôle
  - H de la forme: $H=(P^t, I_{n-k,n-k})$
  - Le récepteur calcule

$$H.\begin{bmatrix}x_0\\\cdot\\\cdot\\x_4\end{bmatrix}=0$$

Les codes linéaires (3)

- **Un code linéaire est cyclique si une permutation d'un mot de code donne un mot de code**
  - Ex: (000,110,101,011)

- **A partir d'un code cyclique on peut construire un code polynômiale, c-à-d on peut trouver un polynôme générateur**
- **Inversement, un polynôme générateur g(x) donne un code cyclique si g(x) est diviseur de X^n+1 (*n* étant le nombre de bits du mot de code)**
  - A partir de g(x), on peut trouver la matrice génératrice et donc les mots de code.
  - Ex: $g(x) = x^3+x^2+1$ pour un code C(7,4)

$$M = \begin{pmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{pmatrix}$$

A faire…

- **Trouver les mots de codes générés par:**

$$M = \begin{pmatrix} 10001 \\ 01010 \\ 00111 \end{pmatrix}$$

  - Le code est-il cyclique?

- **Vérifier que x^3+x^2+1 est diviseur de x^7+1 et donner les mots de code pour un code C(7,4)**

- **Pourquoi un code linéaire contient toujours le mot de code (0,…,0)?**

Code de Hamming linéaire (1)

- Code non systématique utilisant des bits de parité
- Les bits de parité sont insérés à des positions $i=2^n$, $n=0..\infty$
- Les bits de l'information occupent les autres emplacements
- Les bits de parité sont une combinaison linéaire des bits d'information. Les bits de l'information qui contribuent pour un bit de parité à la position 2^j doit posséder 2^j dans sa décomposition binaire.
- Exemple: 3=1+2 et 5=1+4, donc les bit 3 et 5 entrent dans la combinaison linéaire qui définie le bit 1 (qui est un bit de parité).

Code de Hamming linéaire (2)

- **Le code de Hamming peut corriger 1 erreur.**
- **Pour détecter une erreur, le récepteur fait la somme binaire des positions à laquelle il y a un 1 binaire**
  - si la somme est nulle, il n'y a pas eu d'erreurs,
  - si non, la somme donne la position de l'erreur, il suffit de la corriger.

- **Exemple:**

1 2 3 4 5 6 7 8 9 10 11

Soit le lettre H = 0 0 1 1 0 0 1 0 0 0 0 (parité paire)

1 2 3 4 5 6 7 8 9 10 11

La lettre H avec 1 erreur : 0 0 1 1 0 0 1 0 0 1 0

➡la somme binaire modulo-2 de 3, 4, 7 et 10 donne la position de l'erreur!

Cyclic Redondancy Check

- **Code systématique polynômiale sur le principe de la division de polynômes**
- **Utilise les propriétés de l'arithmétique binaire mod-2**
- **Les bits d'information sont représentés sous forme d'un polynôme de degré *N***
- **On utilise un polynôme générateur de degré *n* < *N***
  - le nombre de bits de contrôle qui seront ajoutés dépend directement de *n.*
  - 16 et 32 sont couramment utilisés
- **Soit M(x) le message, G(x) le polynôme générateur**
  - on calcule R(x) tel que $M(x).x^n/G(x) = Q(x)+R(x)/G(x)$
  - on transmet $T(x)=M(x). x^n+R(x)$
  - le récepteur calcule R'(x) tel que T(x)/G(x) = Q'(x)+R'(x)/G(x)
  - si R'(x)=0 alors pas d'erreur.

CRC

Soit 1010 1101 1100 la séquence à transmettre.

Soit $G(x) = X^2+1$, le polynome générateur

En algèbre modulo 2, l'addition et la soustraction sont identiques :

+	0	1
0	0	1
1	1	0

-	0	1
0	0	1
1	1	0

1) transformer la séquence à transmettre en polynome :

$I(x) = 1.x^{11}+0.x^{10}+1.x^9+0.x^8+1.x^7+1.x^6+0.x^5+1.x^4+1.x^3+1.x^2+0.x^1+0.x^0$

$I(x) = x^{11}+x^9+x^7+x^6+x^4+x^3+x^2$

2) multiplier I(x) par x^r où r est le rang de G(x) :

$I(x).x^2 = x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4$

source L. Toutain

Division par G(x)

3) Diviser $I(x).x^r$ par $G(x)$:

$$
\begin{array}{l|l}
x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4 & x^2+1 \\
x^{13}+x^{11} & \hline
\quad x^9+x^8+x^6+x^5+x^4 & x^{11}+x^7+x^6+x^5+x^2+1 \\
\quad x^9+x^7 & \\
\quad\quad x^8+x^7+x^6+x^5+x^4 & \\
\quad\quad x^8+x^6 & \\
\quad\quad\quad x^7+x^5+x^4 & \\
\quad\quad\quad x^7+x^5 & \\
\quad\quad\quad\quad x^4 & \\
\quad\quad\quad\quad x^4+x^2 & \\
\quad\quad\quad\quad\quad x^2 & \\
\quad\quad\quad\quad\quad x^2+1 & \\
\quad\quad\quad\quad\quad\quad 1 &
\end{array}
$$

Le reste vaut 1

source L. Toutain

Vérification au récepteur

4) La séquence transmise est $x^{13}+x^{11}+x^{9}+x^{8}+x^{6}+x^{5}+x^{4}+1$ soit en binaire 10101101110001

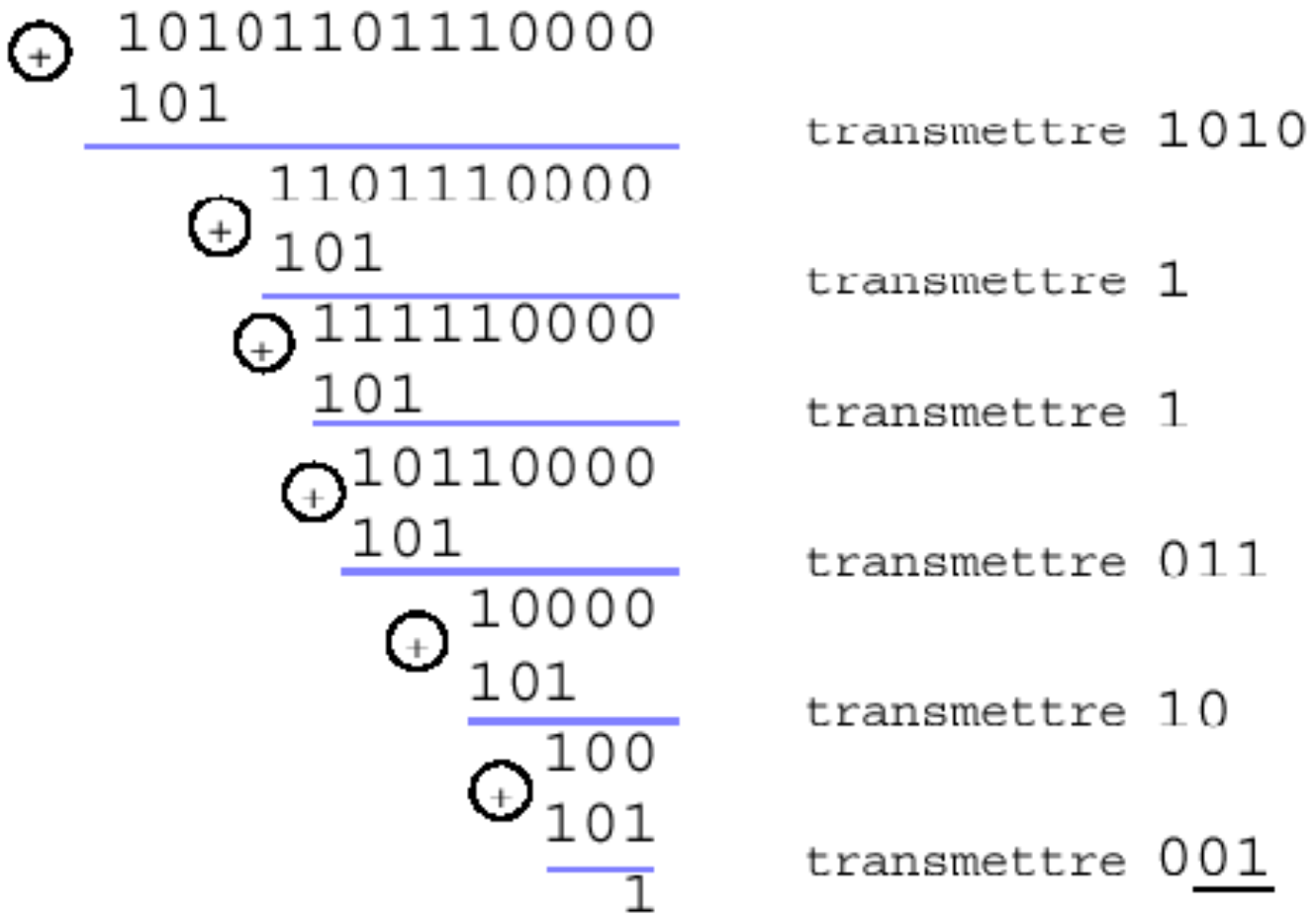
5) Le récepteur effectue la même division sur la séquence reçue

$$
\begin{array}{l|l}
x^{13}+x^{11}+x^{9}+x^{8}+x^{6}+x^{5}+x^{4}+1 & x^{2}+1 \\
\cline{2-2}
x^{13}+x^{11} & x^{11}+x^{7}+x^{6}+x^{5}+x^{2}+1 \\
\cline{1-1}
\quad x^{9}+x^{8}+x^{6}+x^{5}+x^{4}+1 & \\
\quad x^{9}+x^{7} & \\
\cline{1-1}
\quad x^{8}+x^{7}+x^{6}+x^{5}+x^{4}+1 & \\
\quad x^{8}+x^{6} & \\
\cline{1-1}
\quad\quad x^{7}+x^{5}+x^{4}+1 & \\
\quad\quad x^{7}+x^{5} & \\
\cline{1-1}
\quad\quad\quad x^{4}+1 & \\
\quad\quad\quad x^{4}+x^{2} & \\
\cline{1-1}
\quad\quad\quad\quad x^{2}+1 & \\
\quad\quad\quad\quad x^{2}+1 & \\
\cline{1-1}
\quad\quad\quad\quad 0 &
\end{array}
$$

le reste est nul, il n'y a pas d'erreur de transmission

source L. Toutain

Opération à la volée



source L. Toutain

Comment choisit-on un bon CRC?

- **Si erreur, à la place de T(x) on reçoit T(x)+E(x). Chaque bit à 1 dans E(x) correspond à une erreur.**
  - E(x)=00000000100000 ou E(x)= 0000110010000
- **Le récepteur calcule [T(x)+E(x)]/G(x). Comme T(x)/G(x)=0, on calcule en fait E(x)/G(x).**
  - S'il y a une seule erreur, $E(x)=x^i$, si G(x) contient au moins 2 termes, il ne divisera jamais E(x) donc toutes les erreurs simples seront détectées.
  - S'il y a 2 bits isolés, $E(x)=x^i+x^j$, i>j. On peut écrire $E(x)=x^j(1+x^{i-j})$. Si on suppose que G(x) n'est pas divisible par x, une condition suffisante pour détecter toutes les erreurs doubles serait que G(x) ne divise pas x^k+1 pour toutes les valeurs de k jusqu'à i-j (jusqu'à la longueur maximale de trame.
  - On connaît des polynômes (comme $x^{15}+x^{14}+1$) qui ne divisent pas x^k+1 pour k<32768.

Comment choisit-on un bon CRC?

- S'il y a un nombre d'erreurs impairs, alors E(x) contient un nombre impair de 1. On démontre qu'il n'y a pas de polynômes avec un nombre impair de termes factorisable par x+1 (dans le système modulo-2). Donc si on choisit G(x) tel que G(x) soit factorisable par x+1, on pourra détecter toutes trames ayant un nombre impair d'erreurs.
- Un polynôme avec r bits pourra détecter toutes les erreurs en rafales de longueur $k \leq r$, $x^i(x^{k-i}+\ldots+1)$. Si G(x) contient le terme x^0, il ne sera jamais factorisable par x^i et donc si le degré du polynôme dans la parenthèse est inférieur au degré de G(x), alors le reste ne sera jamais nul.
- Si la rafale est de longueur r+1, le reste sera nul si et seulement si la rafale est identique à G(x). Quelle est la probabilité d'une telle éventualité?
- Solution: $1/2^{r-1}$

solution

Quelques exemples de CRC

- **CRC-12: $X^{12}+X^{11}+X^3+X^2+X+1$**
  - utilisé quand la longueur des caractères est de 6 bits
- **CRC-16: $X^{16}+X^{15}+X^2+1$ et CRC-CCITT: $X^{16}+X^{12}+X^5+1$**
  - très utilisés dans les WANs
- **CRC-32: $X^{32}+X^{26}+X^{23}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$**
  - très utilisé dans la plupart des LANs

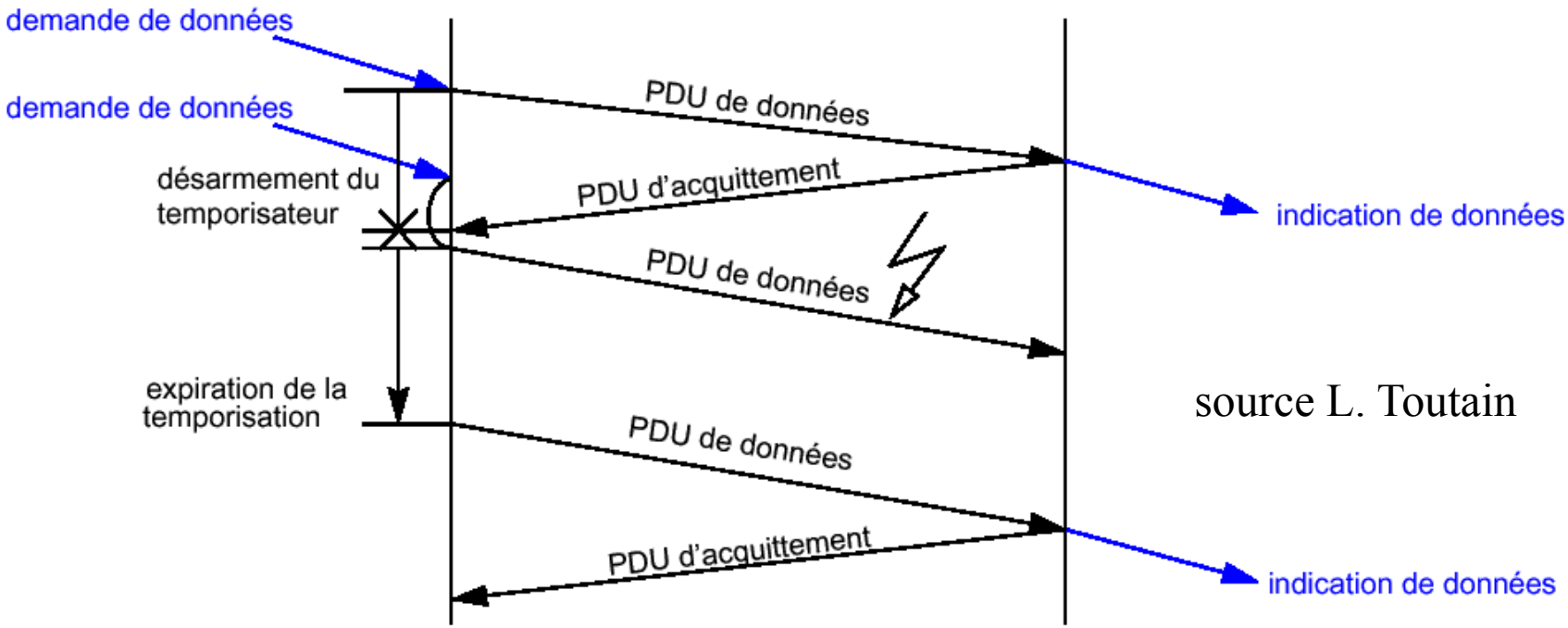
- **Tous 3 sont factorisable par x+1**
- **Le CRC-16 détecte toutes les errreurs isolées et doubles, toutes celles ayant un nombre impair d'erreurs, toutes les rafales de longueur ≤ 16, 99.997 des rafales de 17 erreurs, 99.998 des rafales de 18 et plus erreurs.**

Mécanismes de reprise sur erreurs

- **Gère les pertes et les erreurs non récupérables en demandant une retransmission du(des) trame(s) ➡Automatic Repeat Request (ARQ)**
- **On numérote les trames pour les identifier**
- **Des temporisateurs permettent de gérer les pertes des acquittements.**
- **Idle RQ**
  - initialement pour les blocs de caractères (bytes-oriented)
  - remplacé le plus souvent par Continous RQ
- **Continous RQ**
  - plus efficace que Idle RQ, possède 2 variantes:
  - selective repeat
  - go-back-N

Idle RQ

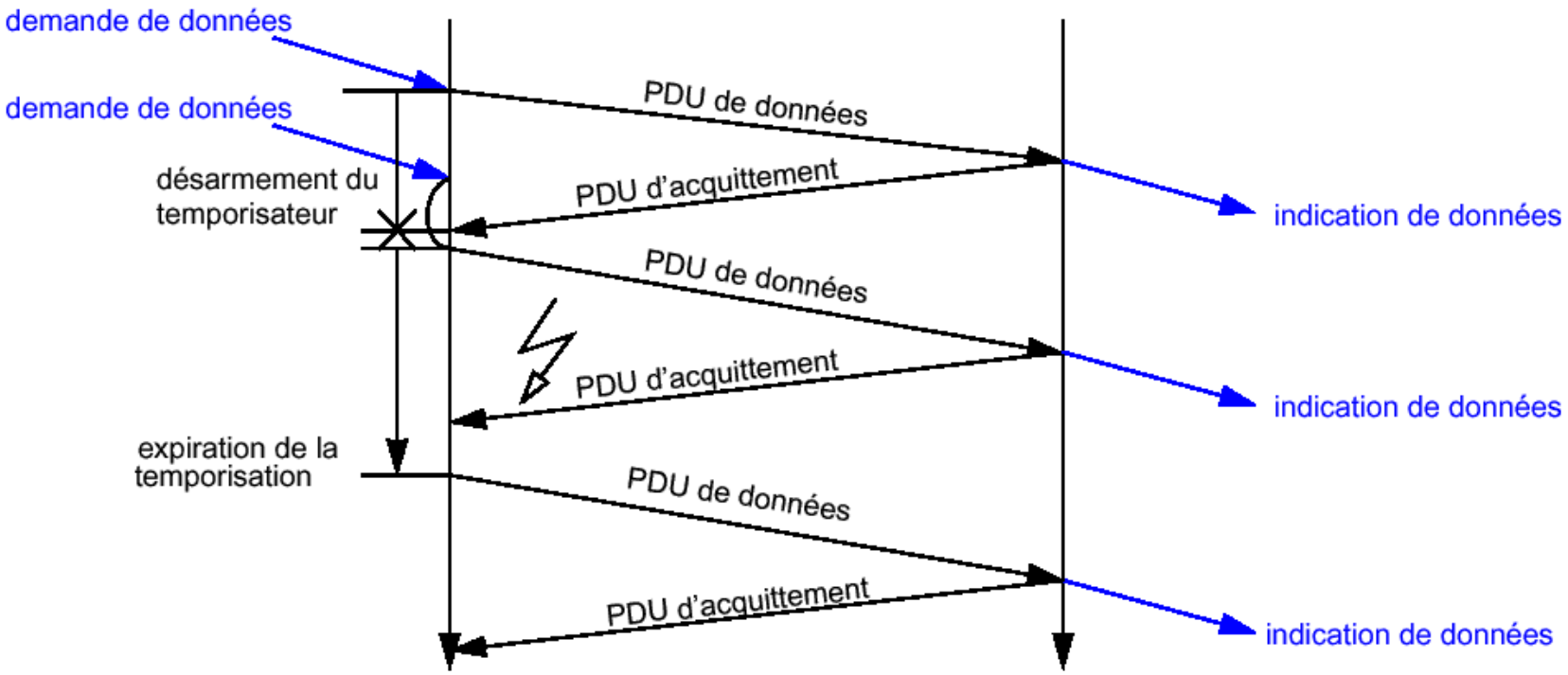
- De l'émetteur, une seule trame de données (I) non-acquitté à la fois (stop-and-go, send-and-wait)
- Le récepteur envoie un ACK pour chaque trame correcte



source L. Toutain

Idle RQ - perte d'un ACK

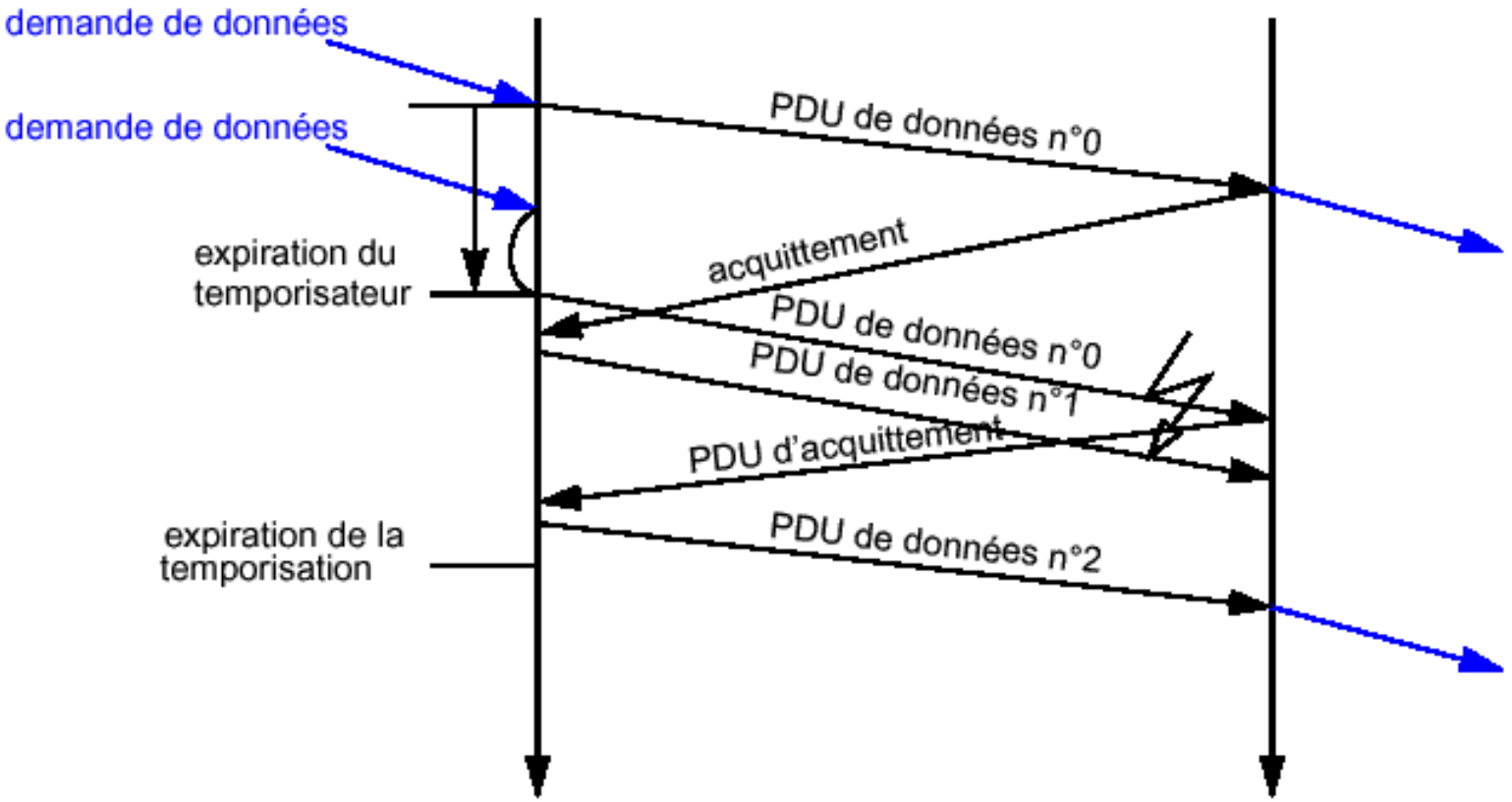
- **Le temporisateur permet de retransmettre automatiquement**



Que se passe t-il si le temporisateur de retransmission est mal dimensionné ?

source L. Toutain

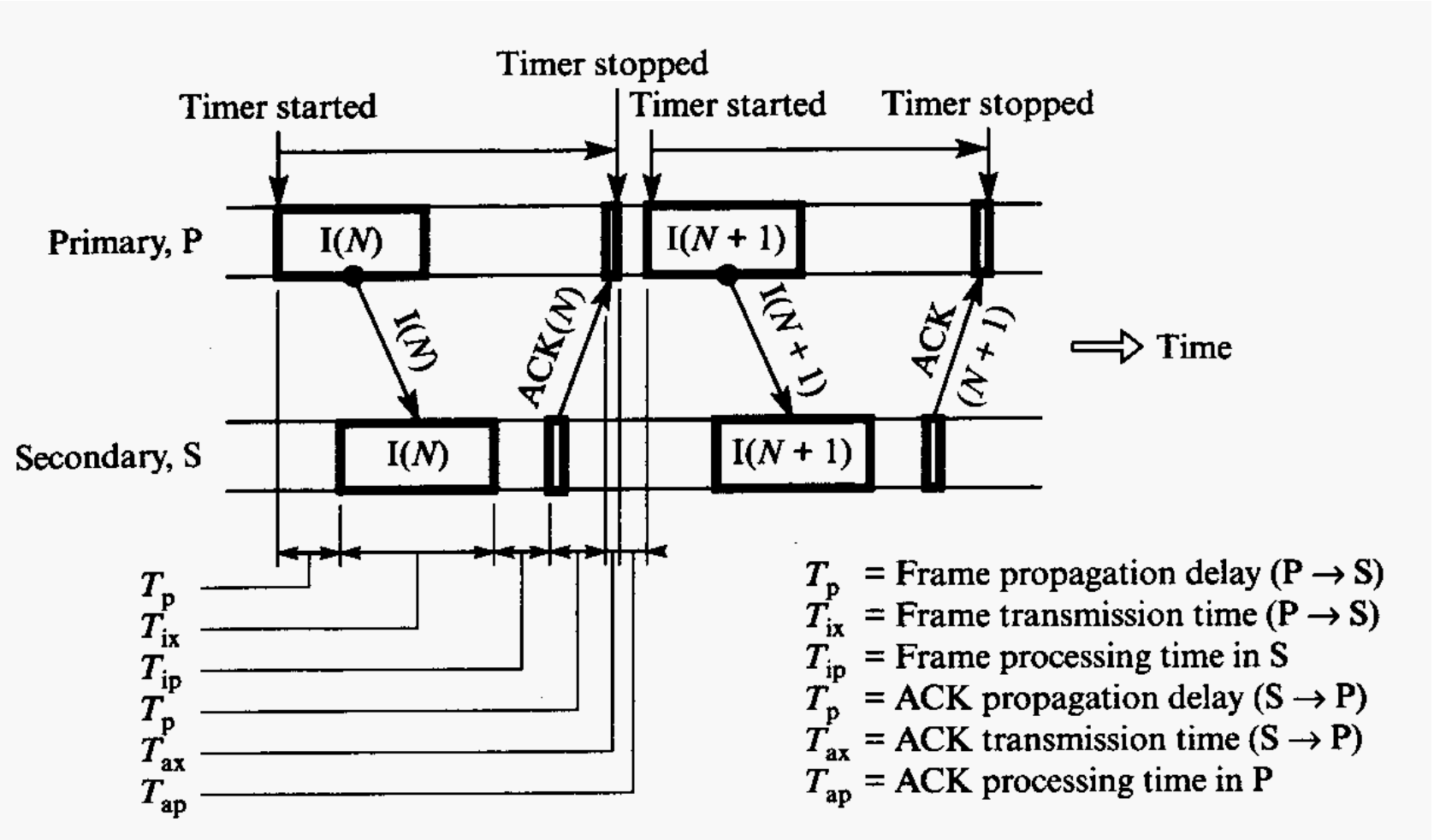
Idle RQ - suite



Quel est le problème ici ? Comment le résoudre?

source L. Toutain

Idle RQ - éléments temporels



Idle RQ - taux d'utilisation du lien parfait

- **Quel est le taux d'utilisation d'un mécanisme comme Idle RQ?**
  - le taux d'utilisation U est défini comme le rapport entre Tx (temps de transmission d'une trame) et Tt (temps d'attente pour un acquittement)
  - on utilisera Tp (temps de propagation sur le lien) et on négligera les temps de traitement (d'une trame et d'un ACK à la réception).

$$U = \frac{T_x}{T_x + 2T_p} = \frac{1}{1 + 2T_p / T_x}$$

Tp/Tx est souvent écrit a

Idle RQ - taux d'utilisation du lien avec erreur

- **On introduit P, le taux d'erreur bit (BER)**
  - on introduira Nr le nombre moyen de retransmissions pour une trame et Ni la taille d'une trame en bit. De même, on négligera les temps de traitement (d'une trame et d'un ACK à la réception).

$$U = \frac{T_x}{N_r T_x + 2 N_r T_p} = \frac{1}{N_r \left(1 + 2T_p / T_x\right)}$$

  - proba qu'une trame soit juste, $(1-P)^{Ni}$
  - Nr = $1/(1-P)^{Ni}$

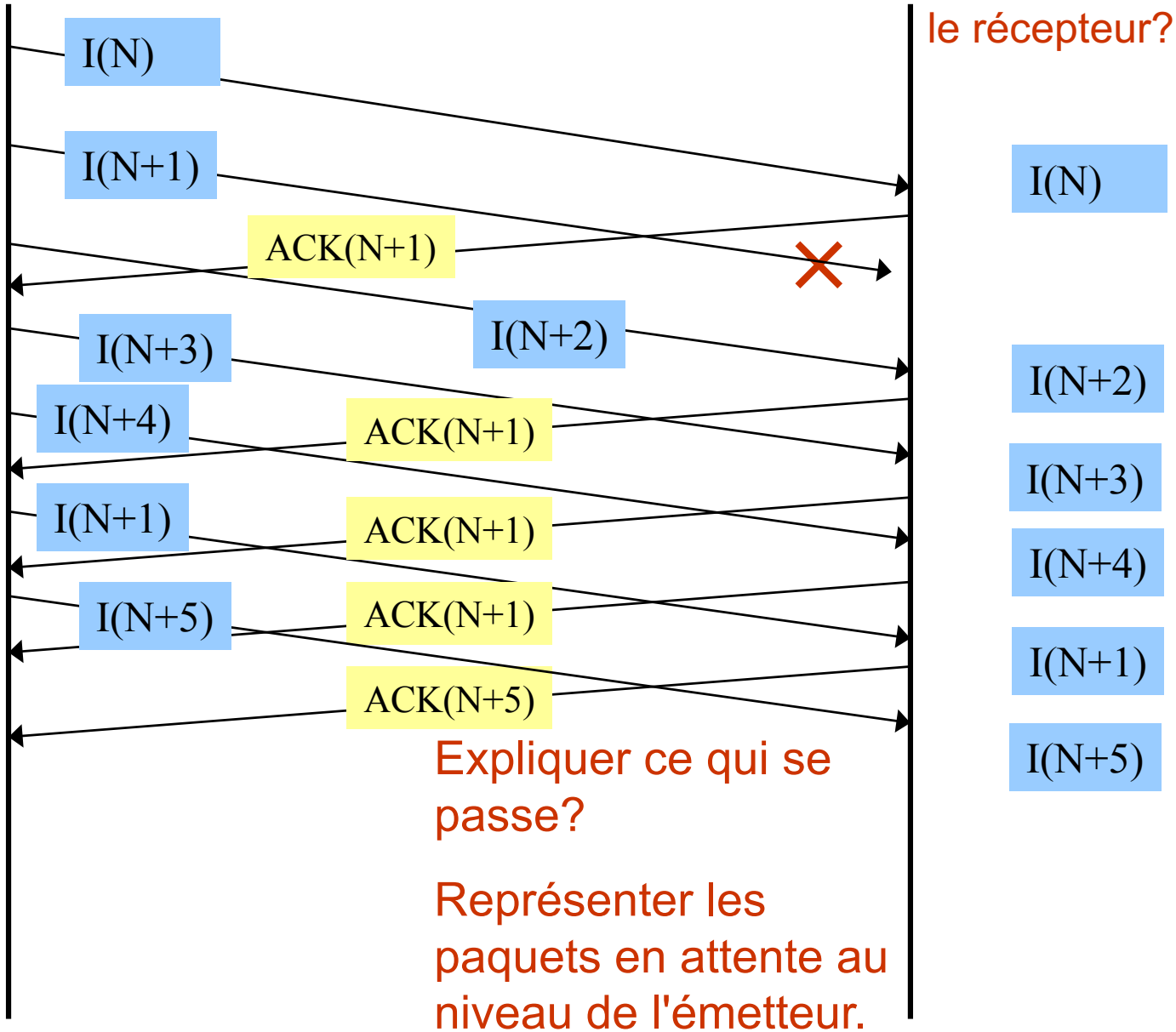
$$U = \frac{(1-P)^{Ni}}{(1+2a)}$$

Continous RQ

- **Permet une meilleur utilisation car l'émetteur peut anticiper et envoyer plusieurs trames sans acquittements.**
- **Nécessite une liaison duplex**
- **Dans le cas le plus simple, l'émetteur peut envoyer plusieurs trames et le récepteur accuse réception de chaque trame**
- **Deux approches**
  - Selective Repeat/Selective Reject
  - Go-Back-N

Selective Repeat

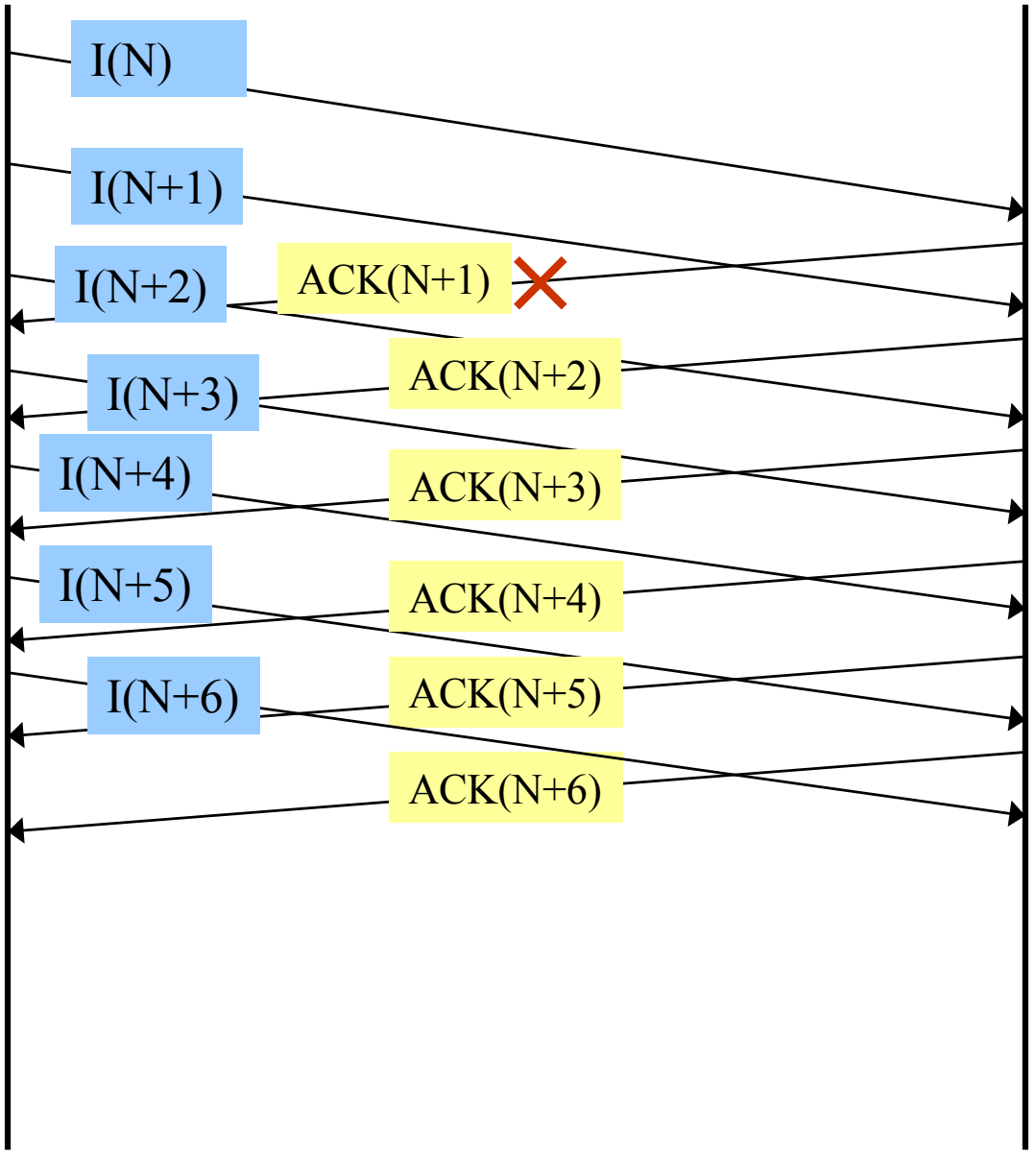
Avantages et inconvénients pour le récepteur?



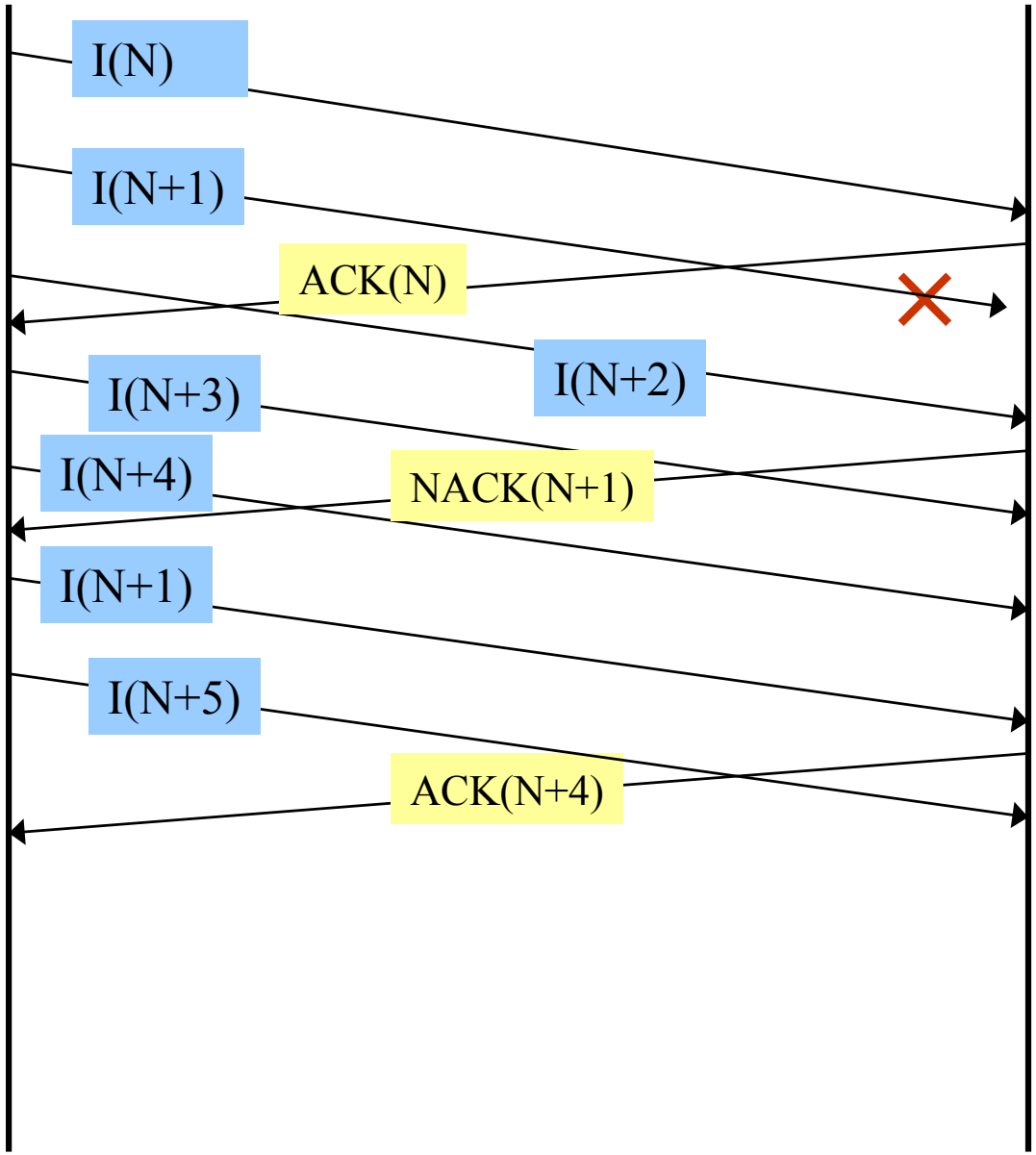
Expliquer ce qui se passe?

Représenter les paquets en attente au niveau de l'émetteur. Que dire?

Selective Repeat - ACK perdu



Selective Reject

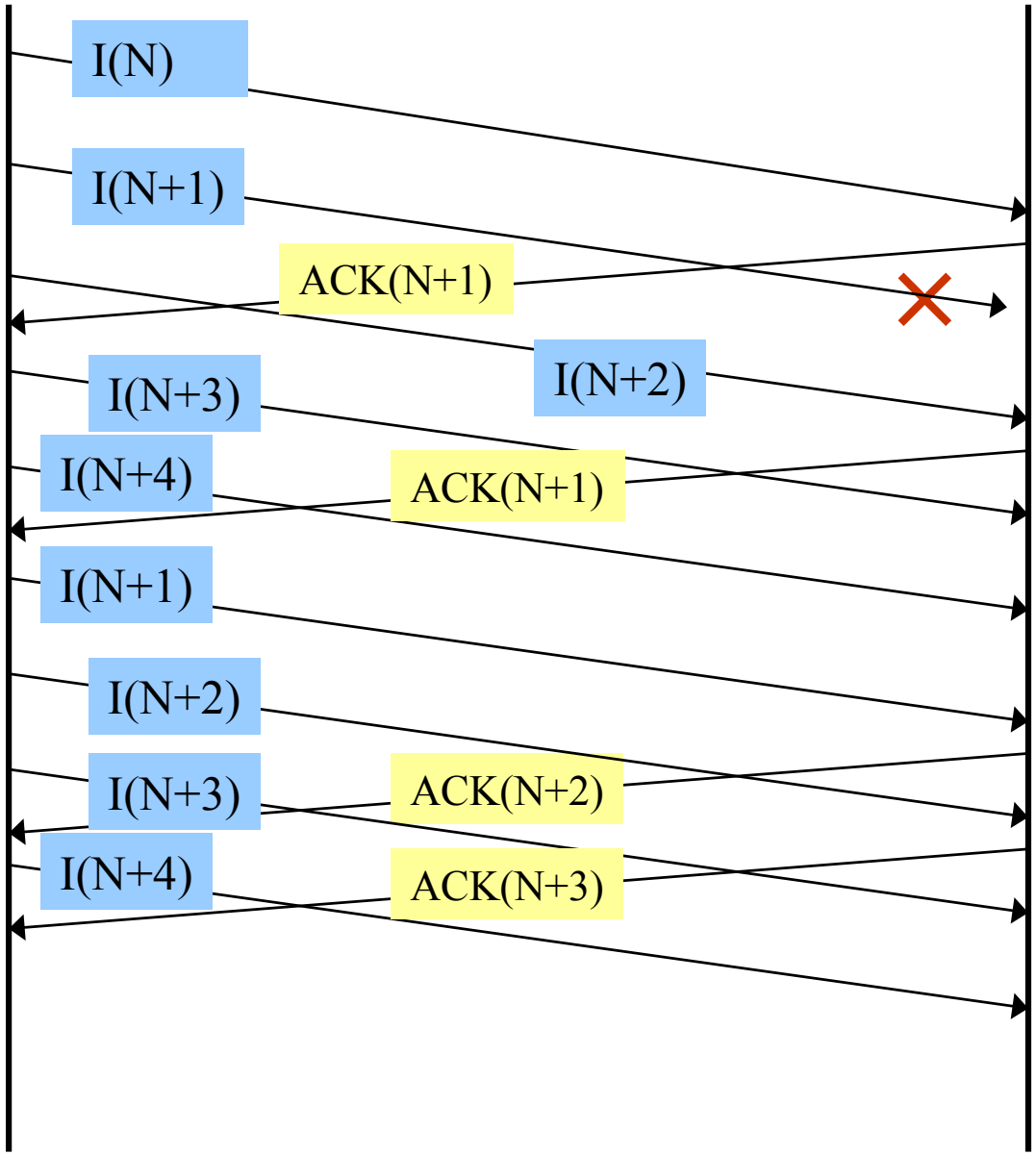


Un ACK(N) acquitte normalement tous les paquets i<N.

Une fois un NAK envoyé, il n'y a pas d'ACK avant d'avoir reçu le paquet perdu. Pourquoi?

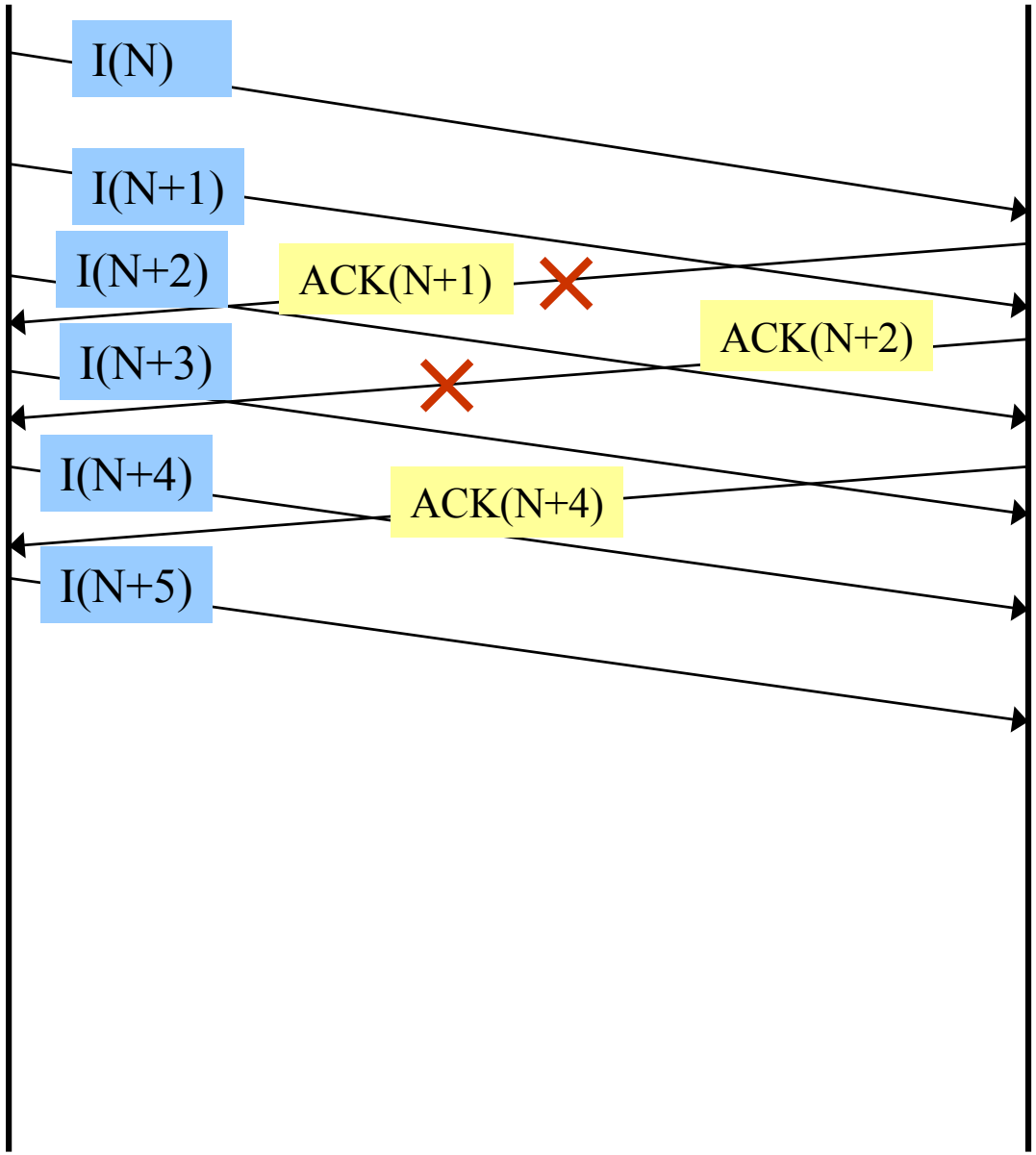
Combien de NAKs peuvent être en attente?

Go-Back-N

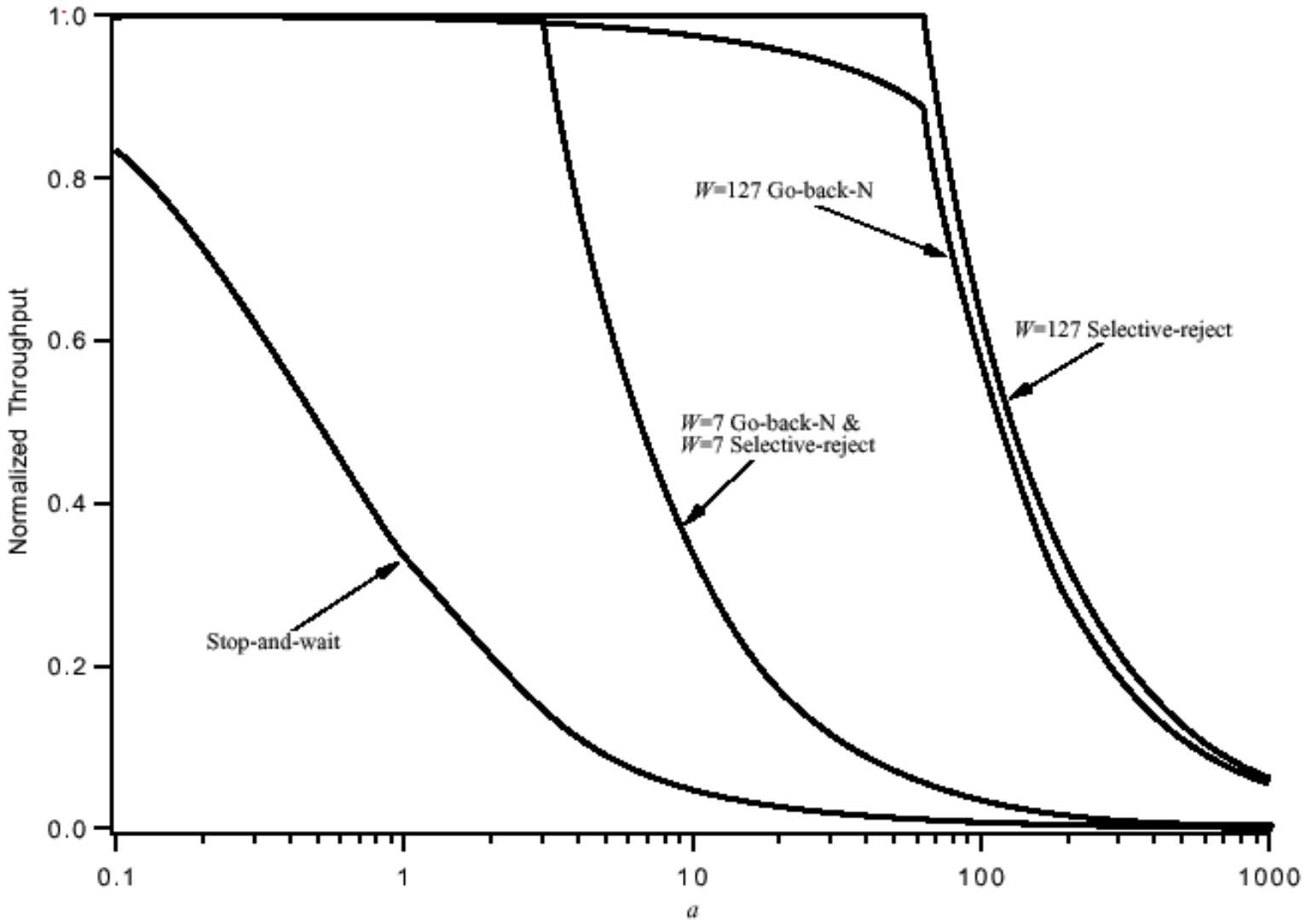


Sur perte d'une trame, on reprend la retransmission à partir de cette trame.

Go-Back-N - pertes de ACK



Débit en fonction de *a=Tp/Tx*



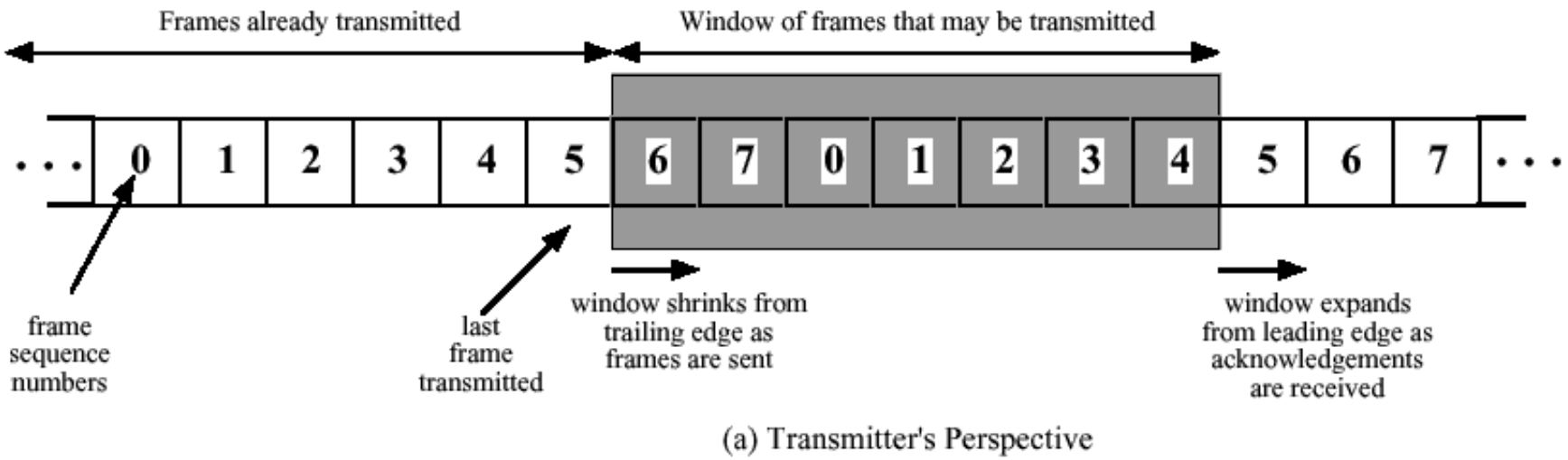
ARQ Throughput as a Function of *a* ($P = 10^{-3}$)

Mécanismes de contrôle de flux

- **Permet de ne pas saturer le récepteur**
- **X-ON/X-OFF**
  - Mécanisme simple et limité
- **Fenêtre coulissante**
  - Mécanisme le plus utilisé

Fenêtre coulissante (Sliding window)

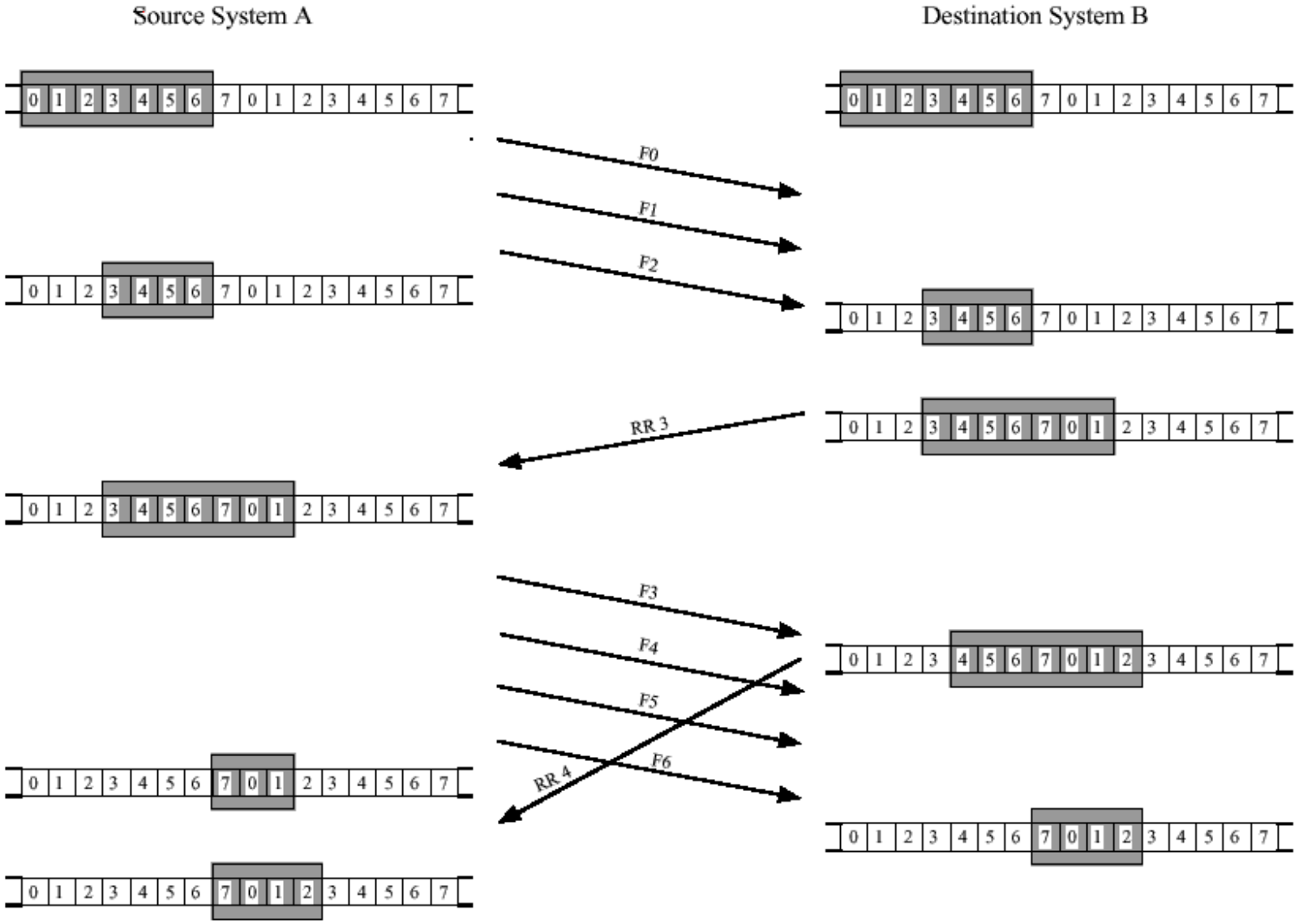
- Permet de ne pas saturer le récepteur



(a) Transmitter's Perspective

➡problème de numérotation

Exemple de fonctionnement



Débit en fonction de *a=Tp/Tx*

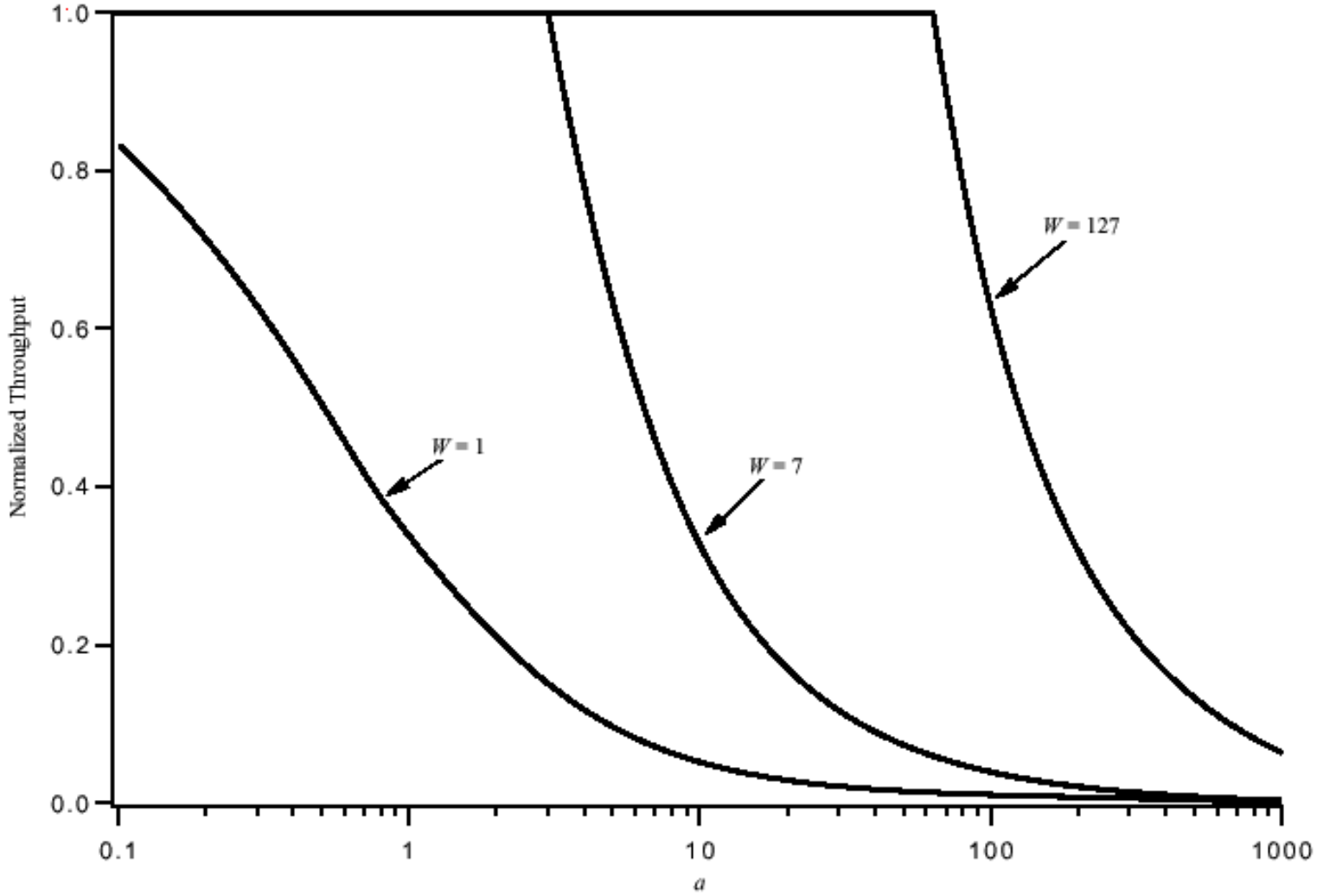


Figure 6.16 Sliding-Window Throughput as a function of *a*

Continous RQ: taux d'utilisation des liens

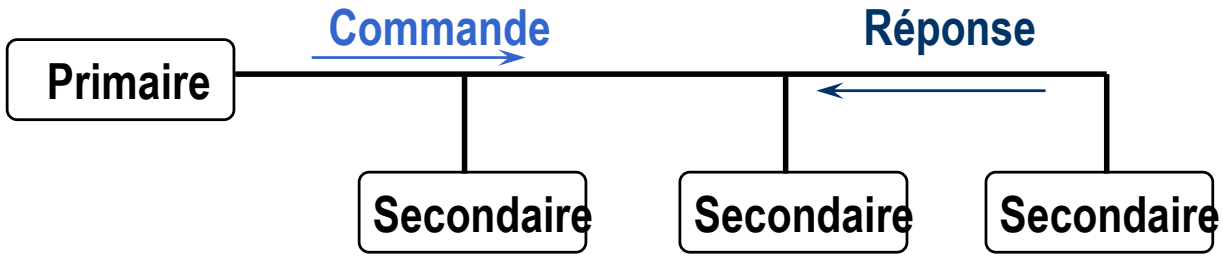
- **En vous basant sur le taux d'utilisation d'un lien avec Idle RQ, proposer une expression pour le taux d'utilisation d'un Continous RQ avec une taille de fenêtre de K**

- **Solution:**
  - on pourra considérer 2 cas, $K \geq 1+2a$ et $K < 1+2a$

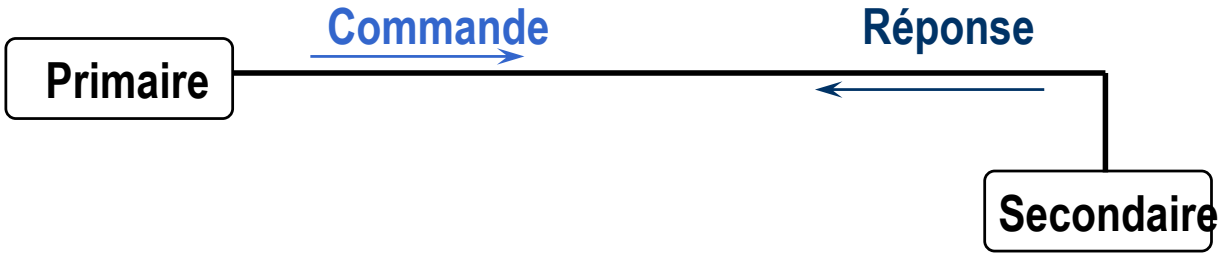
Solution cachée…

Statut des stations

- **Système à commande centralisée**
  - Multipoint



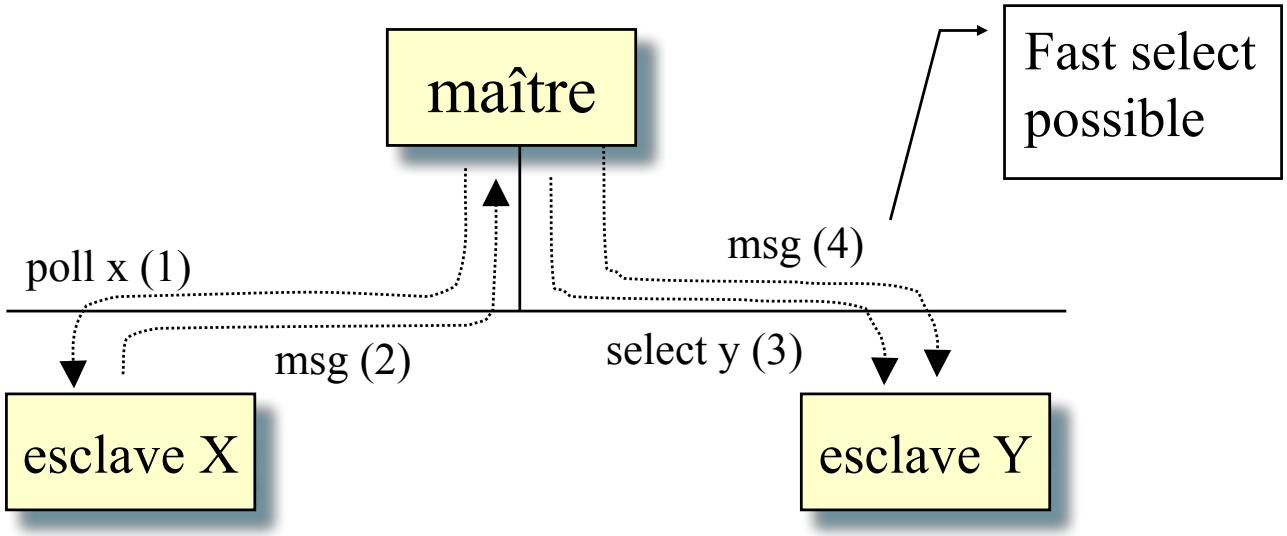
  - Point à point



- **Adresse = station SECONDAIRE**

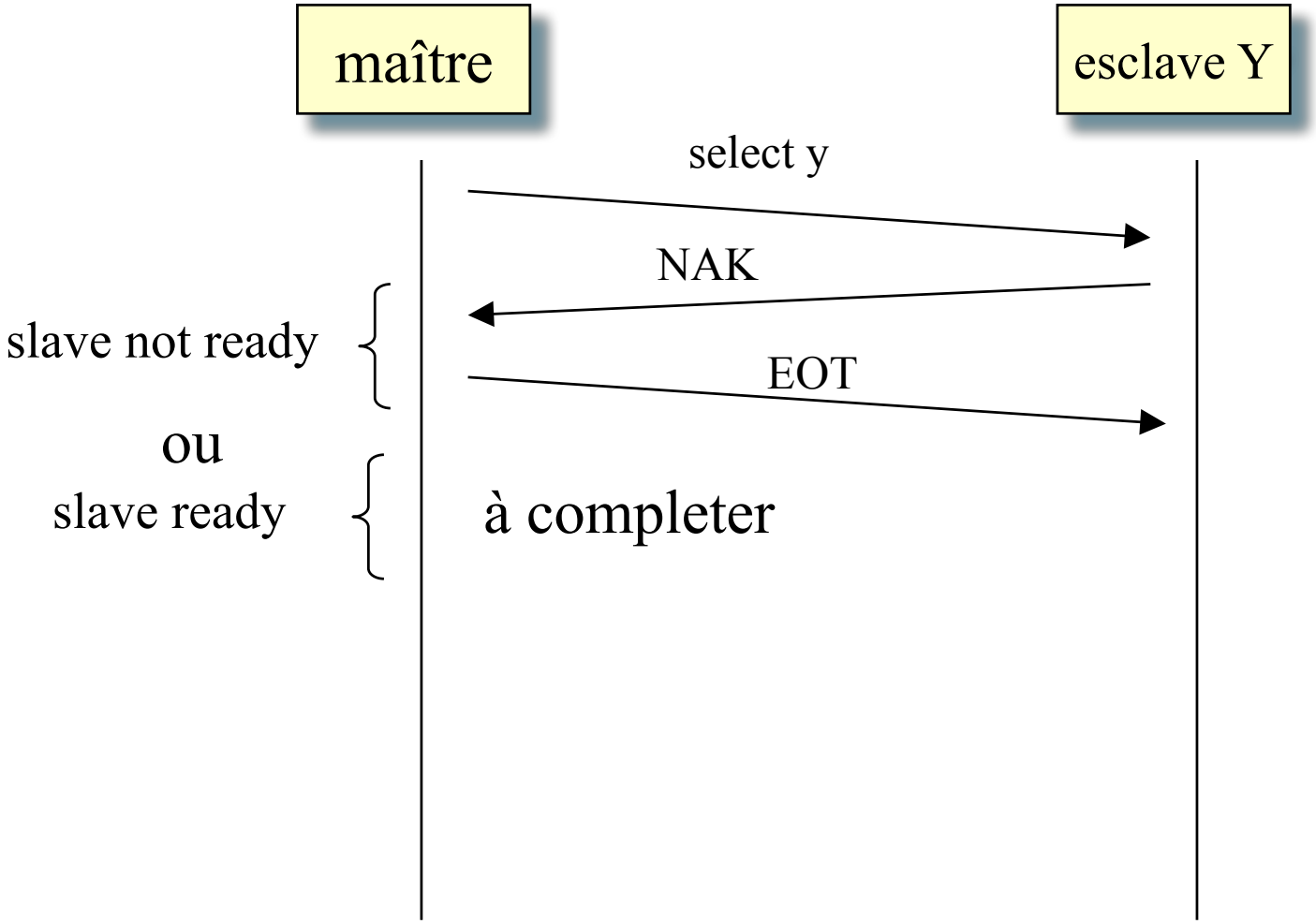
Fonctionnement maître/esclaves

- Une station maître est responsable du déroulement des transmissions sur les liens.
- Le message *poll* permet à un maître d'indiquer à un esclave qu'il peut envoyer tous ses messages en attente.
- Le message *select* permet au maître de demander à l'esclave s'il est prêt à recevoir des données.



A faire…

- **Compléter le chronogramme d'une transmission simple avec la 2ème trame perdue une fois.**



A faire…

- **Performance du protocole en multipoint**
  - lorsqu'il n'y pas de messages à envoyer, le temps pour poller toutes les machines secondaires est faible (Tmin)
  - lorsqu'il y a des messages à envoyer, ce temps (Tavr) augmente et dépend de la fréquence des messages à envoyer (Mr)

$$T_{avr} = \frac{T_{\min}}{1 - M_r T_x}$$

- **Exercice**

  Un protocole BSC est utilisé pour contrôler le flot des messages entre une station maître et 10 stations esclaves sur un lien à diffusion. Le débit du lien, R, est de 10Kbits/s et la longueur moyenne d'un message N est de 1000 bits. Si un *poll* message et son ACK associé font 30 bits et nécessite 1ms de traitement, déterminer le temps moyen entre 2 *polls* pour un esclave lorsque (i) la fréquence est de 1 message/min, (ii) la fréquence est de 6 messages/s.

Solution…

- Temps pour transmettre une trame Tx=N/R=100ms
- Temps pour transmettre un poll et un ACK: 30/R=3ms
- Temps pour poller un esclave = 3ms + 1ms = 4ms
- Temps min pour poller les 10 esclaves: Tmin=40ms

- De $T_{avr}=\frac{T_{\min}}{1-M_r T_x}$
- (i) Mr=10^{-3}/60 msg par ms➡ $T_{avr}=\frac{40}{1-\frac{10^{-3}}{60}.100}=40ms$
- (ii) Mr=6.10^{-3} msg par ms➡

$$T_{avr}=\frac{40}{1-6.10^{-3}.100}=\frac{40}{0.4}=100ms$$

HDLC, LAPB, LAPD, LLC

Autres protocoles dérivés ou non

Flag	Address	Control	Information	FCS	Flag
8	8n	8 or 16	variable	16 or 32	8

(a) HDLC, LAPB

Flag	Address	Control	Information	FCS	Flag
8	16	16*	variable	16	8

(b) LAPD

MAC control	Dest. MAC address	Source MAC address	DSAP	SSAP	LLC control	Info.	FCS
variable	16 or 48	16 or 48	8	8	16*	variable	32

(c) LLC/MAC

* = 16-bit control field (7-bit sequence numbers) for I- and S- frames; 8 bits for U-frames

Autres protocoles dérivés ou non

Flag	Address	Control	Information	FCS	Flag
8	16 to 32	16*	variable	16	8

(d) LAPF (control)

Flag	Address	Information	FCS	Flag
8	16 to 32	variable	16	8

(e) LAPF (core)

Generic flow control	Virtual path identifier	Virtual channel identifier	Control bits	Header error control	Information
4	8	16	4	8	384

(f) ATM

* = 16-bit control field (7-bit sequence numbers) for I- and S- frames; 8 bits for U-frames

Récapitulatif