II -1-3 Increasing complexity protocols

(a) Protocol 1 " Without error and flow control"
(b) Protocol 2 "Send and wait"
(c) Protocol 3 " alternate bit "
(d) Protocol 4 " With sliding window and orderly reception"
(e) Protocol 5 " With sliding window and selective rejet"

I-1-3 PROTOCOLES DE COMPLEXITE CROISSANTE

Ces protocoles apportent des solutions de complexités croissantes aux principaux problèmes posés à savoir :

- Contrôle d'erreur.
- Contrôle de flux.
- Respect de la causalité (livraison en séquence).

Les solutions sont ensuite Codées en langage ADA

Les protocoles suivants seront examinés :

- Protocole 1 "Sans contrôle d'erreur et de flux"
- Protocole 2 "Envoyer et attendre"
- Protocole 3 "Bit alterné"
- Protocole 4 "A fenêtre glissante et réception ordonnée"
- Protocole 5 "A fenêtre glissante et rejet sélectif"

(a)**Protocole 1 "Sans contrôle d'erreur et de flux" :**

Hypothèses de travail :

1- Le code ne décrit que la transmission dans un seul sens :
⇒ Solution proposée unidirectionnelle.
2- La couche réseau du récepteur est toujours prête à recevoir :

- Les temps de calcul induits par le déroulement du protocole sont négligeables.
- On dispose de la mémoire nécessaire pour stocker les messages (tampons) chez l'émetteur comme chez le récepteur.

Autre présentation : le contrôle de flux assuré dans un autre niveau (les pertes de trames au niveau liaison par saturation des tampons sont négligeables).

⇒**Pas de contrôle de flux**.

3- Le canal de communication est parfait ("presque"):
Les pertes de trames sont négligeables sur le support ou le contrôle d'erreur est prévu ailleurs.

⇒**Pas de contrôle d'erreur**.

Nature de la solution

- Solution de base d'un protocole sans connexion qui se contente d'acheminer des trames et laisse aux niveaux supérieurs toutes les tâches.
- Mise en place de la programmation pour les solutions plus complexes.

Programmation du Protocole 1 :

```
--
-- Déclarations
--
-- Zone de données utilisateur (paquet réseau) par exemple 128 octets.
Type paquet is array( integer range 1..128 ) of character ;
-- type de trame de niveau liaison utilisée : rien que le paquet.
Type trame is record
        Info :paquet ;
End record ;
-- type événement en entrée (seulement arrivée de trame )
Type type-événement = (arrivée_trame ) ;
--
-- Procédure exécutée par l'émetteur
--
procédure émetteur_1 is
s          : trame;                          -- Trame en émission
tampon : paquet;                             -- Paquet à émettre
begin
        loop
        recevoir_couche_réseau (tampon);     -- Un tampon à envoyer
        s.info := tampon ;                   -- Préparer une trame
        envoyer_couche_physique (s);         -- La faire émettre
        endloop                              -- Boucle infinie
end émetteur_1;
--
-- Procédure exécutée par le récepteur
--
Procédure récepteur_1 is
événement   : Type_Evénement ;               -- Evénement à traiter
r              : trame ;                     -- Trame en réception
begin
        loop
        attendre (événement) ;               -- Attendre une arrivée
        recevoir_couche_physique (r);        -- Prendre trame arrivée
        envoyer_couche_réseau (r.info) ;     -- passer à l'utilisateur
        endloop                              -- Boucle infinie
end récepteur_1;
```

(b) Protocole 2 "Envoyer et attendre" ("Stop and Wait") :

Hypothèse de travail

- La solution ne décrit qu'un seul sens de communication. Elle utilise une voie de retour pour des trame de service. $\Rightarrow$ Solution unidirectionnelle

- Les erreurs de transmission sont négligées.

⇒ Pas de contrôle d'erreur

- Solution au problème de contrôle de flux.
  - **Freiner l'émetteur** pour ne pas saturer le récepteur. L'émetteur doit émettre des trames à une vitesse au plus égale à la vitesse à laquelle le récepteur retire les données .
  - **Solution minimum de rétroaction sur l'émetteur :**
    - Le récepteur informe l'émetteur qu'il peut **accepter une nouvelle trame** en envoyant une trame de service (unité protocolaire) ne contenant aucune donnée.
    - Cette trame est en fait un **crédit** (**CDT**) (d'une unité) pour émettre une nouvelle trame.
    - L'émetteur **doit attendre d'avoir un crédit** avant d'envoyer une trame.

Nature de la solution

- première solution au problème de contrôle de flux.

Programmation du protocole 2

Les variations par rapport au code précédent sont en italique

```
--
-- Déclarations globales (pas de changement)
--
Type paquet is array (integer range 1..128 ) of character ;
Type trame is record
info :paquet ;
end record ;
type Type_Evénement = (Arrivée_Trame) ;
--
-- Procédure exécutée par l'émetteur
procédure émetteur_2 is
événement : Type_Evènement ;          -- Un événement à traiter
s         : trame;                    -- Trame en émission
tampon  : paquet;                     -- Paquet à émettre
begin
        loop
        recevoir_couche_réseau (tampon);   -- Un tampon à envoyer
        s.info := tampon ;                 -- Préparer une trame
        envoyer_couche_physique (s);       -- La faire émettre
attendre(événement) ;                 -- Attendre un crédit
        endloop                            -- Boucle infinie
end émetteur_2;
--
-- Procédure exécutée par le récepteur
--
Procédure récepteur_2 is
événement   : Type_Evénement ;        -- Evénement à traiter
r           : trame ;                 -- Trame en réception
s           : trame ;                 -- Une trame de crédit
begin
        loop
        attendre (événement) ;             -- Attendre arrivée de trame
        recevoir_couche_physique (r);      -- Prendre trame arrivée
        envoyer_couche_réseau (r.info) ;   -- passer à l'utilisateur
envoyer_couche_physique(s) ;          -- Envoyer le crédit
        endloop                            -- Boucle infinie
end récepteur_2;
```

(c) Protocole 3 "Bit alterné" ("ABP'' Alternate Bit Protocol'') (PAR, ''Positive Acknowledgement with Retry'')

Hypothèse de travail

- Solution unidirectionnelle
  Une communication décrite sur une voie bidirectionnelle.
- Solution au contrôle de flux.
  Protocole 2 avec arrêt et attente
- Solution au contrôle d'erreur.
  - Le canal est **bruité** (perte de messages) .
  - Solution de base avec **acquittement positif, délai de garde, identifications des messages.**

Utilisation d'un **code détecteur d'erreur.**
Acquittement positif si trame correcte.
Délai de garde en vue de retransmission si erreur.
Identification des trames par un numéro de séquence.

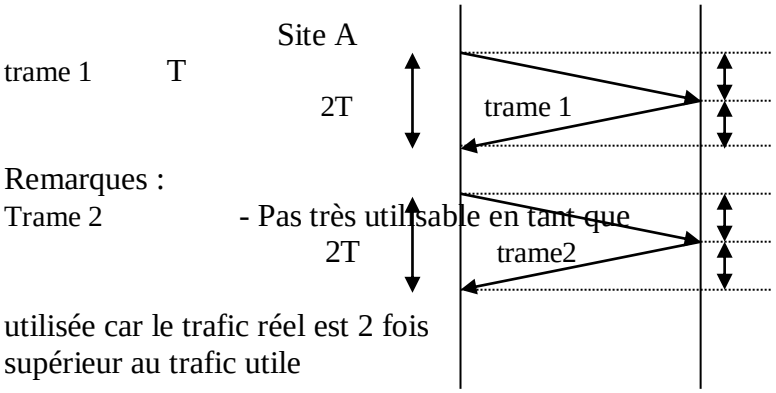
Nature de la solution

- Fournir une **première solution simple** aux problèmes de contrôle d'erreur, de flux et de séquence.
- Possibilité de **nombreuses variantes** dans les stratégies de solution.

c-1) Elaboration d'une solution aux problèmes de flux, d'erreurs et de séquence :

Traitement des erreurs et de contrôle de flux en mode "echoplex" qui est la méthode la plus ancienne qui consiste à gérer les terminaux clavier écran en mode caractère.

Le récepteur retourne une copie de la trame émise (en fait un caractère). L'émetteur compare la trame émise et la trame retournée (en fait contrôle visuel)



Remarques :
- Pas très utilisable en tant que protocole programmable.
- La voie physique est très mal utilisée car le trafic réel est 2 fois supérieur au trafic utile

c-2) Elaboration de la solution du bit alterné :

c-2-1) <u>Acquittements</u> :
(accusé de réception) ("Acknowledgment")

- Acquittements positifs -

Une trame de service indiquant la bonne réception d'une trame de données est Appelée **acquittement positif** ("Positive acknowledgment")

Utilisation

Règle 1 : Une trame n'est **acquittée positivement** que si elle est reçue correctement (code détecteur correct)

Règle 2 : toute trame **correcte doit être acquittée positivement** afin que l'émetteur ne la retransmette plus

Remarques

- **Stratégie de reprise :** En acquittement positif la reprise sur erreur est plutôt confiée à l'émetteur qui doit s'assurer qu'une trame a bien été reçue avant de poursuivre.
- **Acquittements positifs et crédits :** Le premier a une signification dans le contrôle d'erreur alors que le second sert dans le contrôle de flux.

Une trame unique (baptisée acquittement "ACK") est souvent utilisée (exemple dans le protocole PAR) à double sens pour signifier :

- **Dernière trame correcte** (acquittement positif)
- **acceptation d'une autre trame** (crédit d'1 trame).

- Acquittements négatifs -

Une trame de service indiquant la mauvaise réception d'une trame de donnée est appelée **acquittement négatif** (NAK "Negative Acknowledgment").

Utilisation

Règle 1 : Une trame n'est acquittable négativement que si **le destinataire est certain de ne pas l'avoir reçue correctement** alors qu'elle a été émise.

Signal indiquant l'arrivée d'une trame en erreur.
Absence de la trame dans une suite numérotée.

Règle 2 : La signification de l'envoi d'un acquittement négatif est donc celle d'une **demande de retransmission.**

Remarques

- Le protocole PAR **n'utilise pas** (à priori) d'acquittements négatifs.
- On peut concevoir de **multiples variantes** de protocoles utilisant plus ou moins mêlées des stratégies d'acquittements négatifs et positifs.

- Des protocoles probabilistes **basés uniquement sur les acquittements négatifs** sont envisageables.
- Avec les acquittements négatifs la stratégie de traitement des erreurs est **plutôt confiée au récepteur** qui doit découvrir **l'absence d'une trame**, en **demande la retransmission** jusqu'à ce que celle-ci **soit bien reçue.**

c-2-2) Délais de garde (temporisateurs, "timers") :

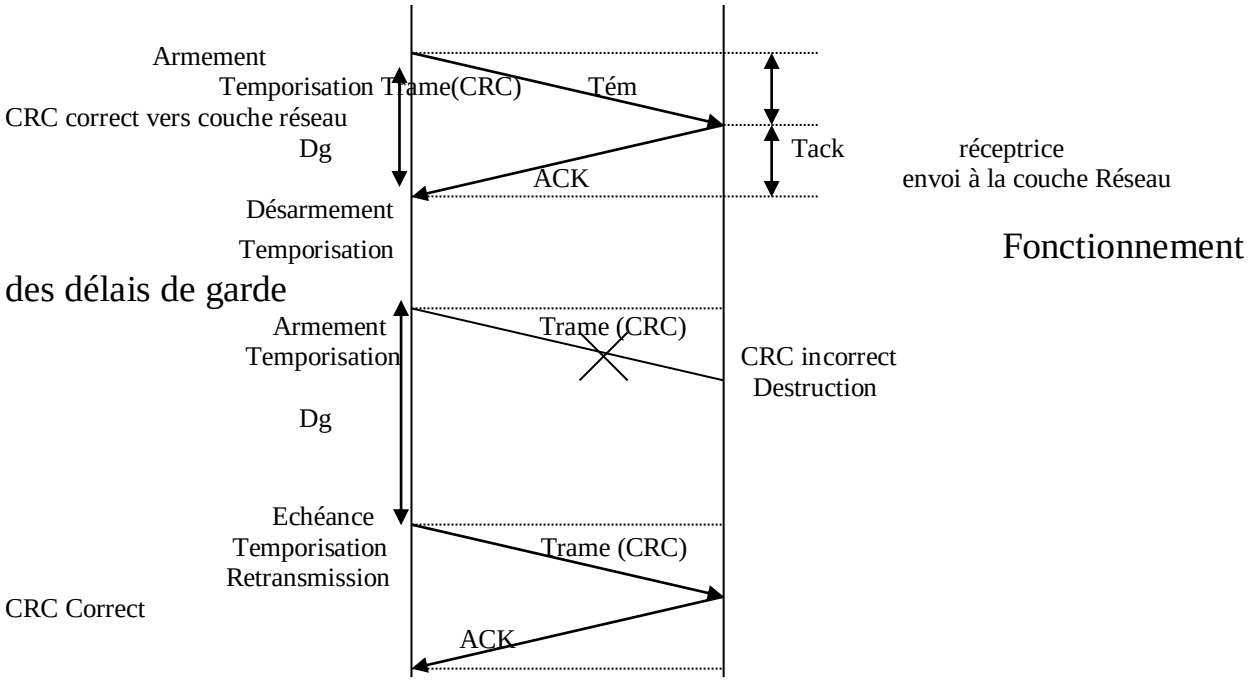
- Nécessité de conserver une **copie d'une trame** pour retransmission en cas d'erreur.
  - Si la trame est en erreur elle est détruite
  - Si l'acquittement positif est en erreur on ne sait jamais si la trame a bien été reçue➔ l'erreur reste inconnue

- On ne peut conserver **indéfiniment** les copies.

- Utilisation d'un **délai de garde** qui "réveille" l'émetteur à échéance et **force la reprise sur erreur.**

- Dans le protocole PAR l'émetteur **retransmet systématiquement la trame** à échéance du délai.

Remarque : L'usage du délai de garde ralentit la reprise sur erreur car le délai de garde (Dg) doit être nettement supérieur aux délais d'émission (Tém) et d'acquittement (Tack) : Dg >>Tém + Tack
=> On reprendrait inutilement beaucoup de trames.



Fonctionnement des délais de garde

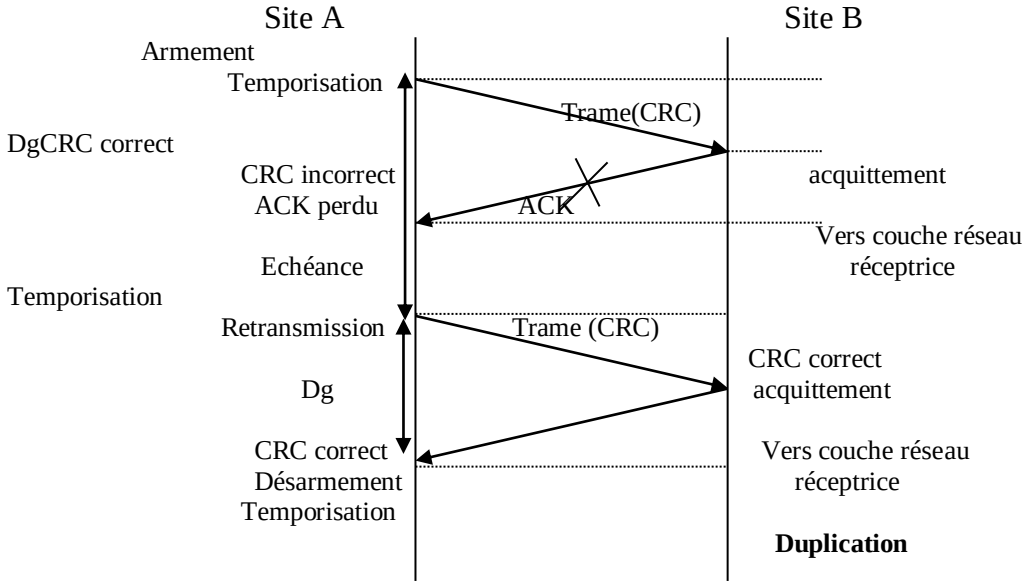
c-2-3) Identification de trames :

Nécessité d'un **numéro de séquence** de trame (identifiant de la trame) pour Eviter les **duplications** et assurer le **respect de la séquence d'émission** (causalité)

Exemple de problème

Une trame est reçue **correctement** mais l'**acquittement positif correspondant se perd…**
La couche liaison du récepteur reçoit **deux fois la trame** puisque l'émetteur réémet.

=>**Duplication non détectée** (non respect de la séquence)

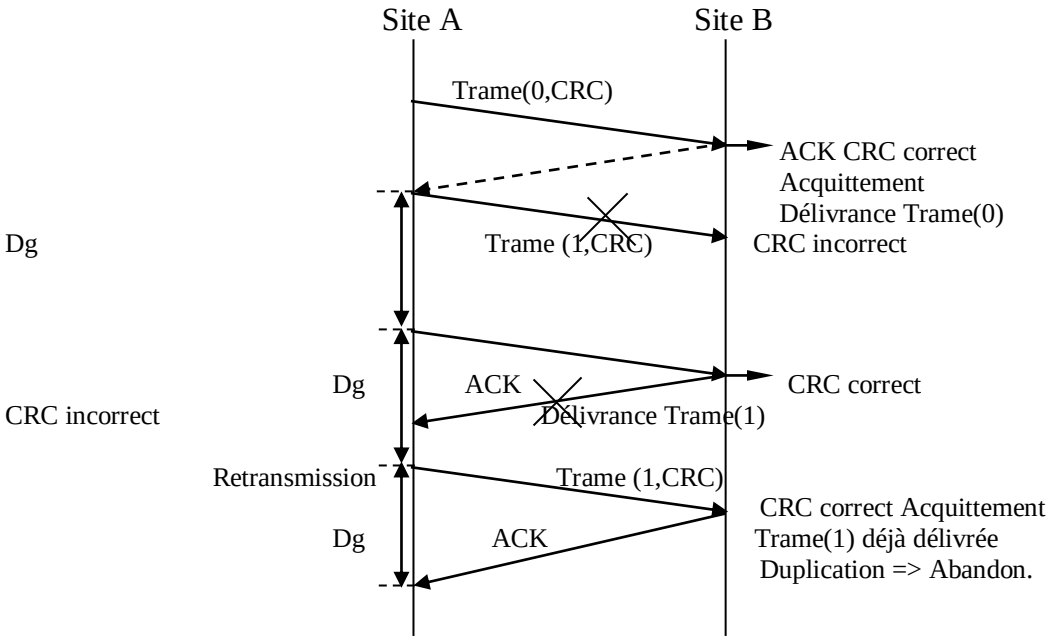


Remarque :

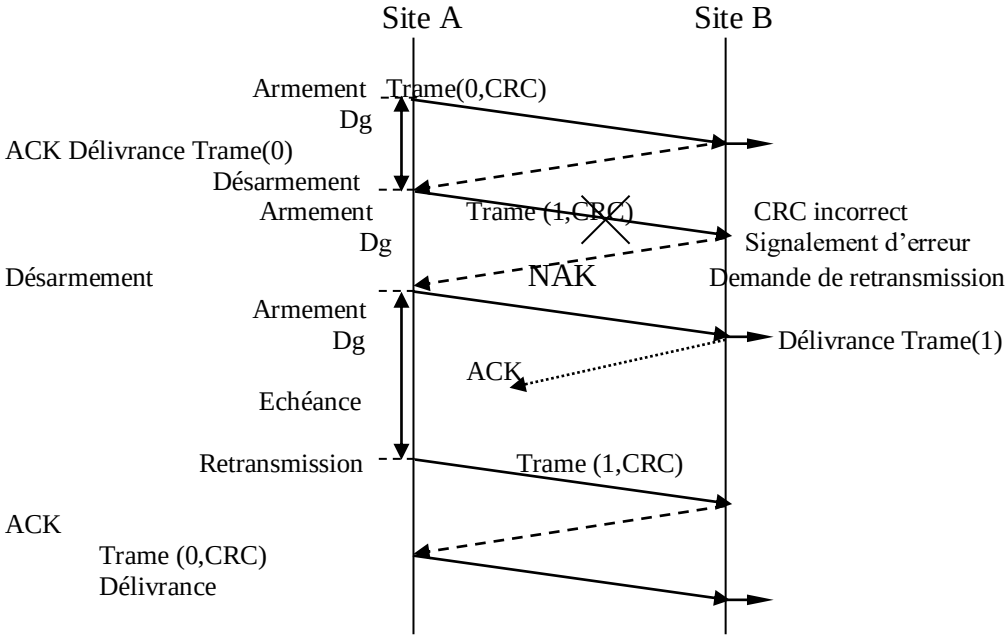
Dans le protocole PAR une numérotation sur un bit suffit pour distinguer une trame de la suivante (protocole de bit alterné).

Exemple d'échange dans le protocole PAR

- une trame correcte.
- Une trame incorrecte.
- Une perte d'acquittement positif.
- Une retransmission réussie.



Variante du protocole PAR :
Utilisation des acquittements négatifs



Remarques :

- Les acquittements négatifs **ne sont pas indispensables** car le fonctionnement de base est fourni par les acquittements positifs, les temporisateurs et les numéros de séquence.
- Les acquittements négatifs servent ici à **accélérer les retransmissions** en cas d'erreur.

Protocole 3 Programmation du protocole PAR

```
--   Variations par rapport au code précédent en italique
--
--  Déclarations globales
--type numéro de séquence d'amplitude 0..maxseq
maxseq : constant :=1 ;
type numéro_ séquence is integer range 0..maxseq ;
type paquet is array (integer range 1..128 ) of character ;
type trame is record
seq : numéro_seq ;
info :paquet ;
end record ;
typeType_Evénement = (Arrivée_Trame, Erreur_Trame, Horloge);
--
-- Procédure exécutée par l'émetteur
--
procédure émetteur_3 is
événement : Type_Evènement ;                    -- Evénement à traiter
s          : trame;                             -- Trame en émission
tampon  : paquet;                               -- Paquet à émettre
Proch_Trame_A_Envoyer : numéro_séquence ;    -- Num prochaine trame émise
begin
Proch_Trame_A_Envoyer := 0 ;                    -- Init pour le premier message
Recevoir_couche_réseau(tampon)                  -- Un tampon est à envoyer
loop
  s.seq := Proch_Trame_A_Envoyer ;          -- Préparer numéro de trame
        s.info := tampon ;                        -- Partie information usager
        envoyer_couche_physique (s);              -- Faire partir la trame
  démarrer_horloge (s.seq) ;                 -- Armer un délai de garde
attendre(événement) ;                          -- Attendre un crédit/ un acquit
if événement = arrivée_Trame then           -- C'est l'acquit attendu
désarmer_horloge(s.seq) ;                   -- Plus besoin d'un réveil
inc(Proch_Trame_A_Envoyer) ;              -- +1 pour le prochain message
recevoir_couche_réseau (tampon) ;       -- Un tampon est à envoyer
endif
    -- Cas d'une retombée de garde : on ne fait rien donc on réémet
endloop                                         -- Boucle infinie
end émetteur_3;
--
```

```
--
-- Procédure exécutée par le récepteur
--
Procédure récepteur_3 is
événement   : Type_Evénement ;          -- Evénement à traiter
r           : trame ;                   -- Trame en réception
s           : trame ;                   -- Une trame de crédit
Trame_Attendue : numéro_séquence        -- Numéro de séquence de
                                        -- prochaine trame à recevoir
begin
Trame_Attendue := 0 ;                   -- Init attente de la trame 0
loop
        attendre (événement) ;          -- Attendre arrivée de trame
-- Deux cas possibles : la trame est correcte ou en erreur
if événement = Arrivée_Trame then       -- Cas d'une trame correcte
        recevoir_couche_physique (r);   -- Prendre trame arrivée
if r.seq = Trame_Attendue then          -- C'est la bonne trame
envoyer_couche_réseau (r.info) ;  -- La passer à l'utilisateur
inc (Trame_Attendue) :                  -- Préparer trame suivante
endif
      -- Dans tous les cas : en séquence ou pas on doit acquitter
envoyer_couche_physique(s) ;            -- Envoyer le crédit / acquit
endif
-- Dans le cas d'une erreur on ignore la trame reçue
endloop                                 -- Boucle infinie
end récepteur_3;
```

c-2-4) Problèmes du protocole PAR :

(a) Robustesse

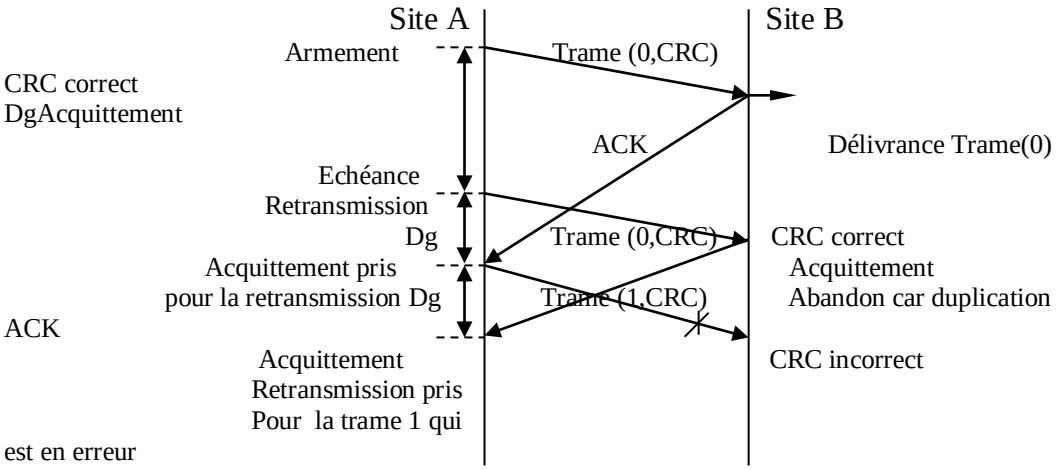
Un protocole correct devrait fonctionner correctement dans toute configuration de :

- **perte de messages** ou **d'acquittements**;
- **réglage des délais de garde** ;
- **délai d'acheminement.**

=> Preuve formelle de protocole.

Rappel : La voie physique ne **déséquence pas** les messages mais le protocole **n'est pas reconfiguré** à chaque fois que l'on change de support ou de débit.

Exemple de problème du PAR : Combinaison d'un délai de garde court et d'une perte de message



Problème : Les acquittements ne sont pas correctement associés aux messages.

Solution : Identifier dans les acquittements le numéro du message acquitté.

- ⇨ Construction de solutions correctes de type PAR.
- ⇨ Protocole à fenêtre glissante de taille 1.

(b) Performances

1- Chaque acquittement positif occupe une trame uniquement dédiée à cette fonction.

=> ▪ Gestion d'une interruption pour chaque acquittement.
Sauvegarde du contexte
Traitement d'IT + Déroutement couche réseau
Restauration du contexte.

=> ▪ Réservation puis libération d'un tampon pour stocker en mémoire chaque acquittement.

Relativement coûteux pour une opération très simple.

2- Si les délais de transmission sont longs, l'attente d'un acquittement est très pénalisante.

Le taux d'utilisation de la voie (notamment les voies satellite) peut devenir très faible et peut tomber à quelques pourcent.

(d) Protocol 4 " With sliding window and orderly reception " :

Objectifs

- Définir un protocole point à point **bidirectionnel simultané** en optimisant les possibilités offertes.

- Protocole qui **corrige les défauts** du protocole PAR.
- **Protocole robuste** qui traite correctement le problème de contrôle d'erreur, de flux, de livraison en séquence :
  - en présence de perte de messages ou d'acquittements
  - pour tout réglage des délais de garde**.**
- **Protocole performant** qui optimise l'utilisation de la voie en fonction :
  - des temps de transmission,
  - de la disponibilité des tampons.

Protocole adaptable de façon performante à différents modes de fonctionnement.

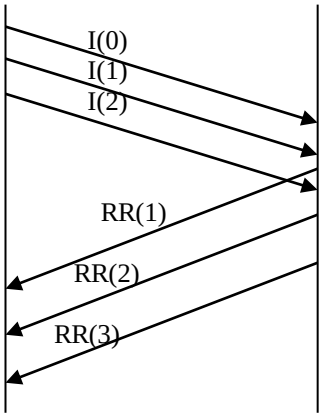
- Le protocole visé reprend les principales options du protocole PAR : l'utilisation des acquittements positifs, des délais de garde, des numéros de séquence de message.
- Protocole présenté en utilisant les notations des protocoles industriels type LAPB.

Amélioration 1 **- Emission en anticipation ("Pipelining")**

- Pour éviter d'être souvent **bloqué en attente** d'acquittement et résoudre le **problème d'inactivité** de la voie si les temps de transmission sont **longs** et les messages **courts**.

- L'**anticipation** des émissions consiste à **ne pas attendre l'acquittement d'une trameavant d'envoyer les trames suivantes**.

Exemple d'émission avec anticipation



I(n) ("Information") : Trame d'information numéro n
RR(n) ("ReceiveReady") : Trame d'acquittement pour n-1

Règles de fonctionnement de l'émission (avec anticipation)

Règle 1 : L'émetteur doit **conserver une copie des trames** jusqu'à réception de l'acquittement correspondant.

⇨ Pour retransmission si les trames ont été bruitées.

Règle 2 : chaque trame est **identifiée par un numéro de séquence**.

Les trames successives sont numérotées circulairement (modulo Maxseq +1) par des entiers successifs.

- ⇨ Pour respecter à la réception l'ordre d'émission.
- ⇨ Pour pouvoir éventuellement détecter des erreurs de transmission par des lacunes dans la suite des numéros reçus.

L'expéditeur maintient une **variable d'état V(s)** qui définit le numéro de la **prochaine trame à émettre.**

Chaque trame est transmise avec un **numéro de séquence en émission N(s)** qui est la valeur courante de V(s) au moment de l'émission.

Règle 3 : Utilisation indispensable d'un ensemble de numéros de séquence de cardinal plus élevé que pour le bit alterné (2 numéros) pour permettre une anticipation réelle. Sur n bits on obtient 2^n numéros différents.

Trames numérotées de 0 à Maxseq = 2^n-1.

- ⇨ Compromis retenus

Maxseq = 7 n=3 Peu de possibilités d'anticipation.
Maxseq = 127 n=7 Encombrement des messages.

Utilisation circulaire des numéros de séquence

- ⇨ Calculs modulo Maxseq+1

Règle 4 : l'anticipation ne peut pas être **autorisée sans limites**.

- ⇨ On n'exercerait aucun **contrôle de flux**.
- ⇨ On ne disposerait pas de la **capacité mémoire suffisante** pour les copies de trames en attente d'acquittement.

Notion de crédit maximum statique (taille maximum de la fenêtre d'anticipation).

Remarque concernant les performances

- En fait l'anticipation revient à **augmenter la longueur** des trames en enchaînant la transmission de plusieurs trames consécutives.
- On **minimise** donc l'importance relative du temps de **propagation aller retour** et on **améliore le taux d'utilisation du canal.**

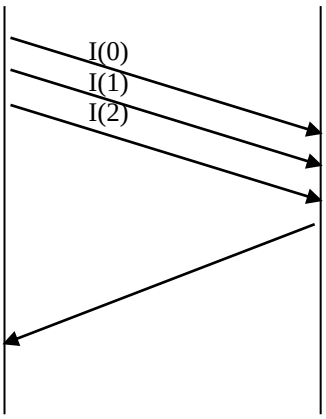
Amélioration 2 – Le regroupement des acquittements

- Il est **inutile et coûteux** d'envoyer **un acquittement pour chaque trame** d'information.

- On peut acquitter plusieurs trames d'information **I** par une seule trame RR d'accusé de réception à condition d'adopter la convention :

Acquittement pour n-1 vaut pour n-2, n-3,… en attente d'acquittement.

Exemple d'émissions avec anticipation et de regroupement des acquittements



RR(3)

Règles de fonctionnement de l'acquittement

Règle 1 : Le récepteur maintient une **variable d'état V(R)** qui désigne le **numéro de séquence de la prochaine trame attendue.**

⇨ Cette variable est incrémentée de 1 chaque fois qu'une trame est reçue en séquence sans erreur.

Règle 2 : La variable de réception V(R) est reportée dans le champ **N(R)** (**le numéro de séquence de réception**) porté dans les acquittements retournés à l'émetteur RR(N(R)).

Règle 3 : Cohérence initiale des numéros de séquence et numéros d'acquittement : N(S) et N(R) = 0 au moment de l'établissement de la liaison.

Règle 4 : On donne la signification suivante à l'acquittement :
RR(N(R)) acquitte toutes les trames en attente d'acquittement **dont le numéro N(S) est inférieur ou égal à N(R)-1.**
Non pas une seule trame dont le numéro serait N(S)=N(R)-1.

Amélioration 3 – L'insertion des acquittements dans les trames d'information (″piggybacking″)

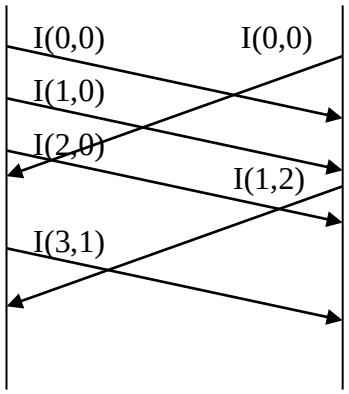
- Insertion en plus du **numéro de séquence** (N(S)) d'un champ **acquittement** (N(R)) dans la partie entête des trames d'information.
  - ⇨ Toute trame d'information devient **un acquittement positif pour des trames du trafic échangé en sens inverse.**

I(N(S) , N(R)) acquitte toutes les trames d'information transmises dans l'autre sens avec des numéros de séquence N(S) inférieur ou égal à N(R)-1

- **L'acquittement inséré coûte quelques bits par trame d'information**
  - ⇨ peu de trames explicites d'acquittement.
  - ⇨ beaucoup plus de possibilités d'acheminer des acquittements.

Sauf si le trafic d'informations est très faible dans un sens : retour à un acquittement explicite.

Exemple d'émissions avec anticipation et acquittements insérés et regroupés



Réalisation des améliorations proposées
Notion de fenêtre en émission

- L'anticipation des émissions introduit **un crédit d'émission**
  - ⇨ Mécanisme d'**optimisation** et de **contrôle de flux.**
- **Limitation** indispensable du crédit
  - ⇨ Allocation à l'émetteur d'un crédit maximum d'émission **We**.

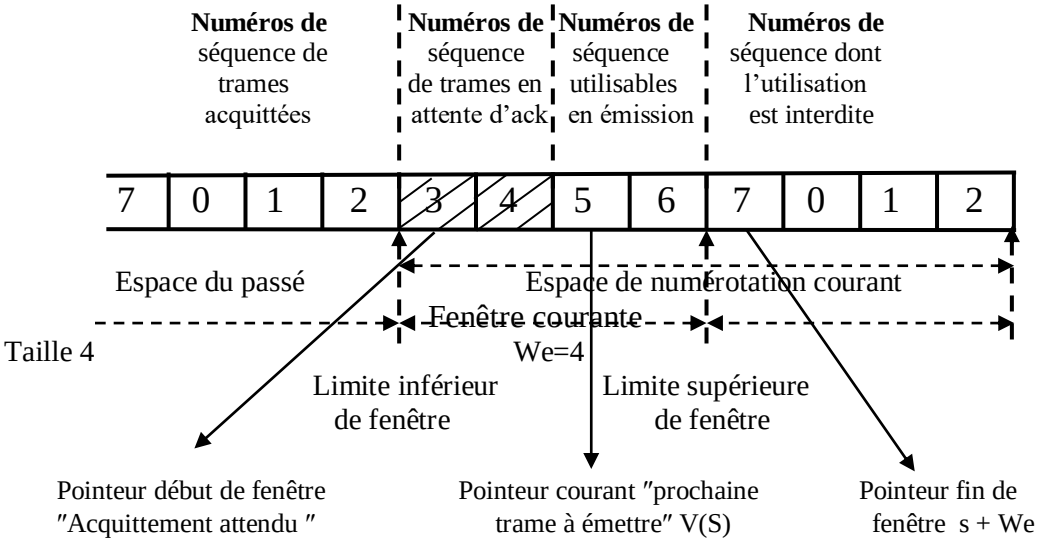
La fenêtre d'émission (″Sender'sWindow″) est l'ensemble des numéros de séquence des trames dont l'émission en anticipation est autorisée.

Numéros des trames d'information en attente d'acquittement.
Numéros de séquence utilisables pour des trames à émettre.

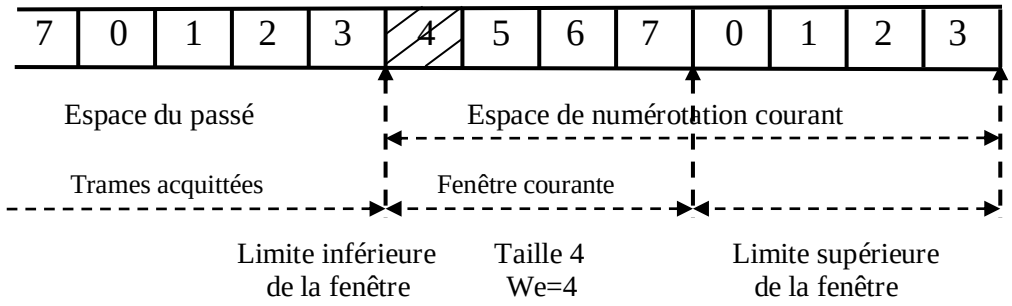
- La fenêtre est définie par $s \leq N(S) < s + We$:
  - s est le numéro de la **plus ancienne trame non acquittée**, qui est la limite inférieure de la fenêtre.
  - S + We est la limite supérieure de la fenêtre, qui est le **numéro de la première trame** dont l'envoi est **interdit.**

- Quand une (ou plusieurs) trames sont acquittées la fenêtre d'émission glisse circulairement vers le haut.
  =>d'où le nom de **fenêtre glissante** (ou coulissante) (″**Sliding windows protocols**″)

Exemple de fenêtre en émission



Fenêtre après glissement (réception RR(4))



- L'émetteur doit disposer des tampons lui permettant de **stocker la fenêtre en émission** (principe d'anticipation).
- Le récepteur **devrait** disposer des tampons lui permettant de ne pas perdre des trames **si l'anticipation est maximum alors que le destinataire final est lent.**

Réalisation des améliorations proposées
Notion de fenêtre en réception

La fenêtre de réception ("Receiver'sWindow") c'est l'ensemble des numéros de séquence des trames que le récepteur est autorisé à recevoir

⇨ Toute trame dont le numéro de séquence correspond à un numéro de la fenêtre de réception est **acceptée.**

⇨ Toute trame dont le numéro de séquence est à l'extérieur de la fenêtre de réception est **détruite.**

- Soit une **trame reçue correctement et dont le numéro de séquence correspond au niveau bas** de la fenêtre en réception :
  - Elle peut-être **délivrée à l'utilisateur** car elle est en séquence (respect de l'ordre d'émission).
  - La fenêtre en réception peut **glisser d'une unité vers le haut.**
  - La trame peut-être **acquittée** vis à vis de l'émetteur.
  - Ces opérations sont réalisées **de façon plus ou moins rapide** sans incidence sur le fonctionnement correct du protocole.
- La fenêtre d'émission et la fenêtre de réception peuvent être de tailles différentes.

Fenêtre en réception dans le protocole 4

- La taille de la fenêtre en réception est égale à 1 :

⇨ le récepteur est obligé de **recevoir** les trames correctement les unes après les autres **exactement dans l'ordre d'émission** (à la limite un seul tampon suffit).

- Quand une trame est **en erreur** le récepteur (qui persiste à ne vouloir qu'une seule trame) **perd toute la série** de trames émises en anticipation par l'émetteur :

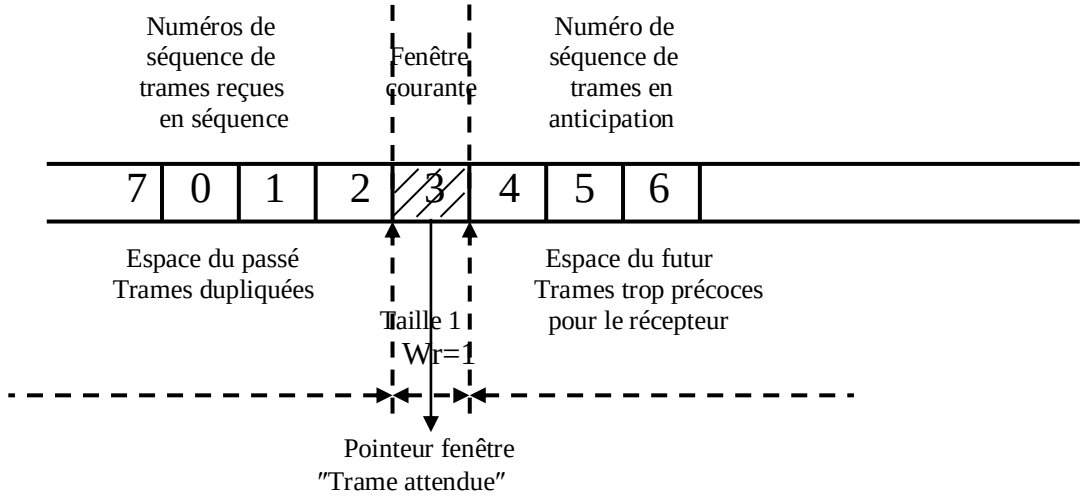
⇨ Effort d'anticipation perdu (optimisation protocole 5)

- L'**émetteur s'aperçoit de la perte :**
  - Stratégie de délai de garde et acquittement positif
    Lorsque le **délai de garde de la trame expire.**
  - Stratégie d'acquittement négatif
    Lorsque le récepteur **constate une lacune** dans la séquence des messages et **demande la retransmission** de toute les trames :

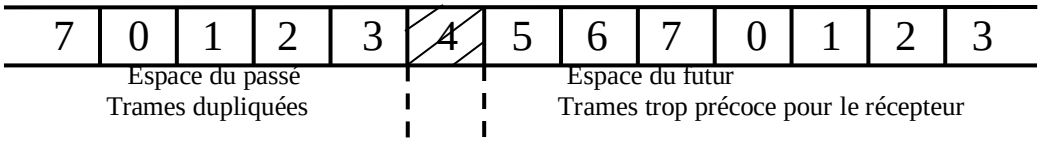
N(S)>N(R)-1

(technique "Go back n" ou de gestion active).

Protocoles à fenêtres glissantes
Exemple de fenêtre en réception de taille1

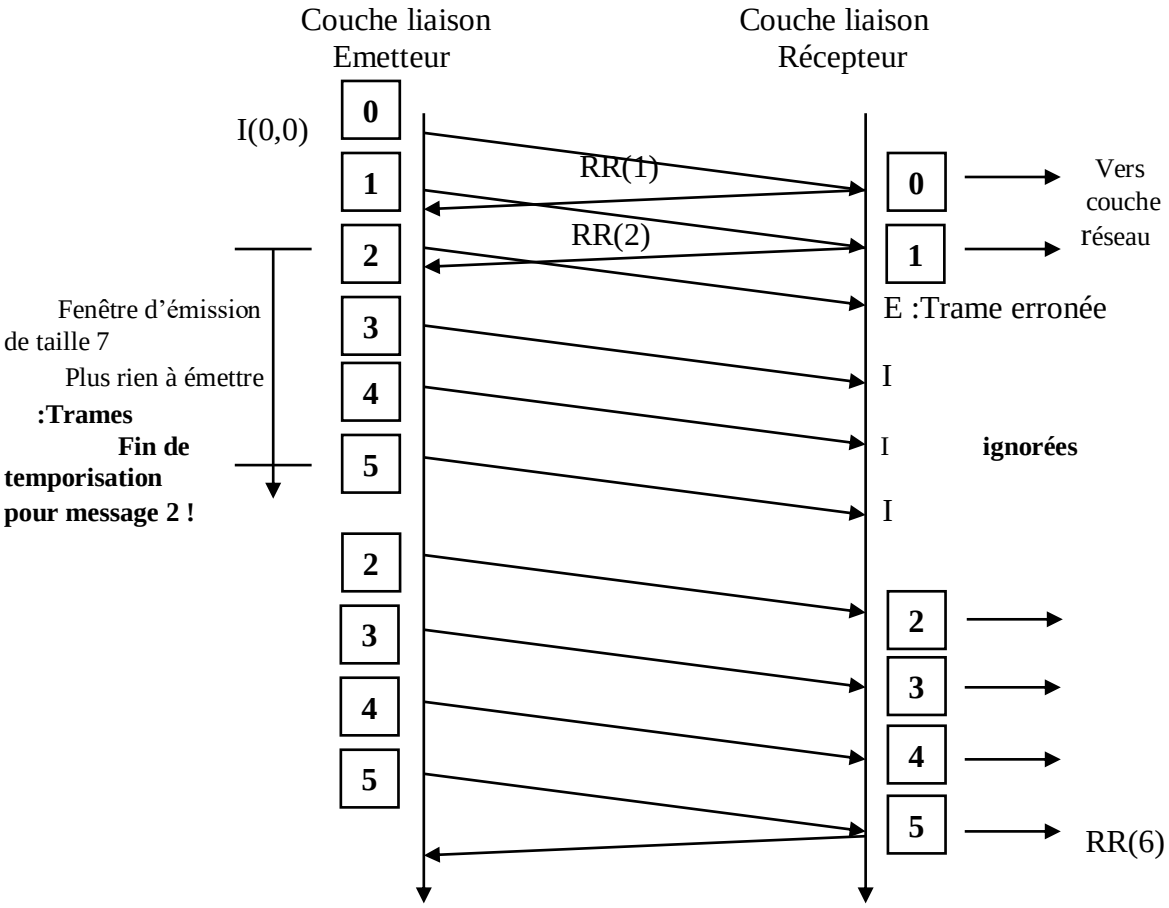


Fenêtre après glissement (réception I(3))



Exemple de fonctionnement du protocole 4

Transmission avec fenêtre d'émission et réception des trames en séquence.



(e) Protocole 5 "A fenêtre glissante et rejet sélectif " :

Objectif

- Chercher à conserver le **bénéfice de l'anticipation** en **cas d'erreur de transmission**.

Solution

- On utilise une fenêtre de réception de **taille supérieur à 1** Wr> 1

- Wr définit la **plage des numéros N(S)** de trames d'informations **acceptables par le destinataire.**

⇨ Le récepteur accepte des trames déséquencées (avec les lacunes dans la numérotation). Il doit donc **gérer pour chaque trame de la fenêtre un booléen indiquant l'arrivée correcte.**

⇨ On reconstitue la séquence par **retransmission sur échéance** de délai de garde ou sur **acquittement négatif.**

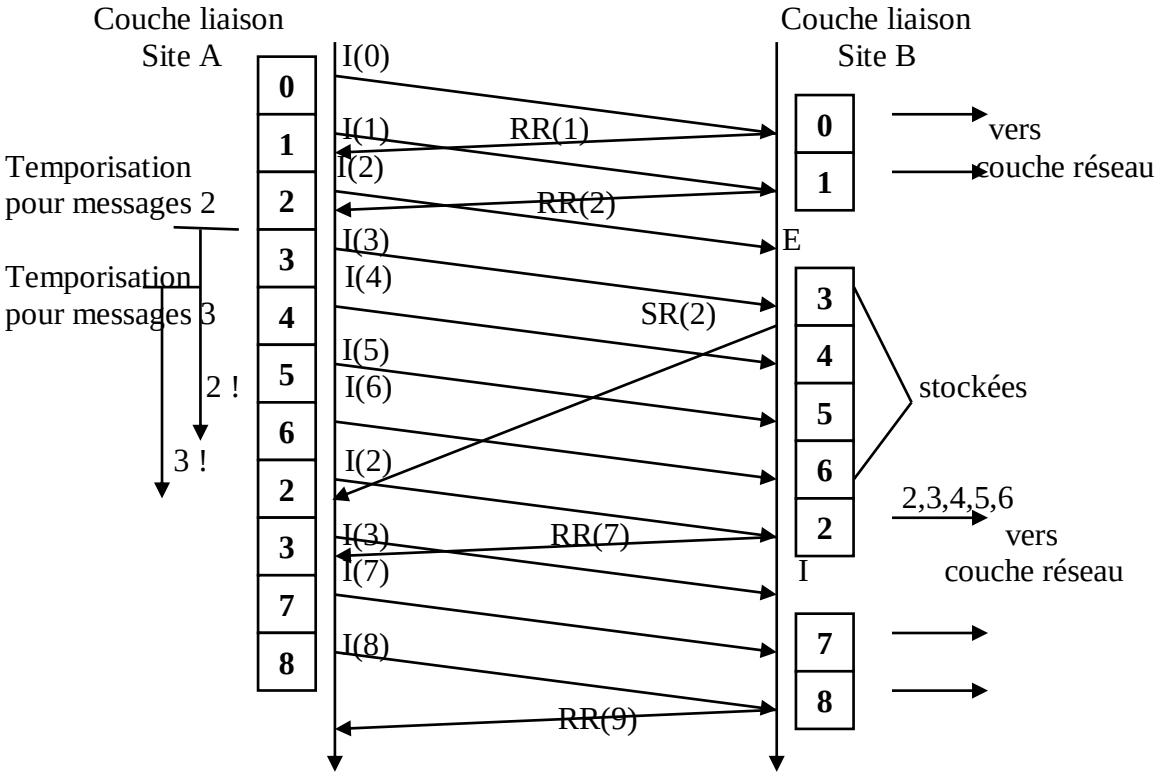
- L'acquittement négatif est baptisé également rejet sélectif (SR(N(R) ″Selective Reject″)

⇨ C'est une **demande de retransmission d'une seule trame d'information en erreur** (de numéro de séquence N(R)), à l'exclusion des suivantes puisque ces dernières ont été en général bien reçues.

Dimensionnement des fenêtres émission et réception

We<Wr : pas très utile car Wr - We tampons en réception ne sont jamais utilisés.
We>Wr : on ne traite pas complètement le problème de perte du bénéfice d'anticipation en cas d'erreur.
We = Wr : le choix le plus rationnel.

Exemple de dialogue en mode rejet sélectif



I-1-4 CONCLUSION : " PROBLEMES GENERAUX DE REALISATION DES PROTOCOLES DE LIAISON EN POINT A POINT"

- **Existence de solutions satisfaisantes** au problème de liaison compte tenu des hypothèses retenues lors de leur conception (années 1970)

- **Résoudre le problème du contrôle d'erreur** pour des voies physiques assez fortement bruitées (10^{-5} par bit)
  - ⇨ Amener la liaison à un taux d'erreur de l'ordre de 10^{-12}.

- **Résoudre le problème de contrôle de flux** pour les voies physiques au débit assez lent et des taux de transmission assez faible.

- **Renouvellement du problème**

Arrivée des réseaux à haut débit sur fibre optique avec des caractéristiques de comportement très différents.

- **Tendance**

La conception des couches physiques, liaison, réseau ne traite que les problèmes de délimitation, routage, congestion mais pas de contrôle d'erreur.

- **Les contrôles d'erreur et de flux sont reportés au niveau transport :** les solutions sont analogues à celles de la liaison avec néanmoins des différences non négligeables de traitement.