CHAPITRE 05

GESTION DE LIAISON

But du chapitre

A la fin de ce chapitre l'étudiant connaitra les différentes fonctionnalités de la couche liaison, entre autres :

- La structuration des données en trame : découpage orienté caractère, orienté bit, etc.
- Le contrôle d'erreurs : les codes VRC, LRC, CRC, etc.
- Les techniques de retransmission de trames : les protocoles « Stop & Go » et « Go-back-N »
- Le fonctionnement du protocole HDLC :
  - Les différents types de trames : données, contrôle, non numéroté.
  - L'établissement et la terminaison de connexion.
  - La transmission de trames avec ou sans contrôle de flux.

1 Introduction

Nous avons vu qu'une donnée est transmise d'un équipement à un autre sous forme de signaux par la couche physique. La donnée transmise peut subir des altérations au cours de son acheminement à cause de différents facteurs extérieurs (ex. le bruit) qui peuvent changer les caractéristiques physiques du signal. On dit qu'une communication a eu lieu entre un équipement source et un autre de destination si les données transmises par la source arrivent telle qu'elles sont au destinataire. Pour ce faire, différentes techniques de gestion de communication sont à mettre en place. Cette gestion de communication doit être assurée tout au long du chemin reliant la source à la destination. Comme un chemin est constitué d'un ensemble de points représentant les équipements par lesquels la donnée passe pour arriver à sa destination finale, donc la gestion de communication peut être assurée de bout en bout (de la source à la destination) ou pour chaque liaison point à point. Cette dernière est la fonctionnalité de la couche liaison.

2 La couche liaison

La fonctionnalité principale de la couche liaison est d'assurer une transmission *point à point* entre deux *systèmes adjacents.* Donc elle assure une *approche locale* de la gestion de communication et non de bout en bout. On dit que cette couche fournit un *canal virtuel fiable* à la couche supérieure (la couche réseau) en assurant une transmission de données entre un émetteur et un récepteur *sans erreurs* et avec détection des débuts et fins des trames constituant ces données.

Comme le transfert de données dans un réseau local est assuré par les fonctionnalités des deux premières couches ; physique et liaison (voir chapitre 6), donc on dit que la couche liaison joue un rôle d'interface entre les couches supérieures et les différentes technologies des réseaux locaux (notez que)

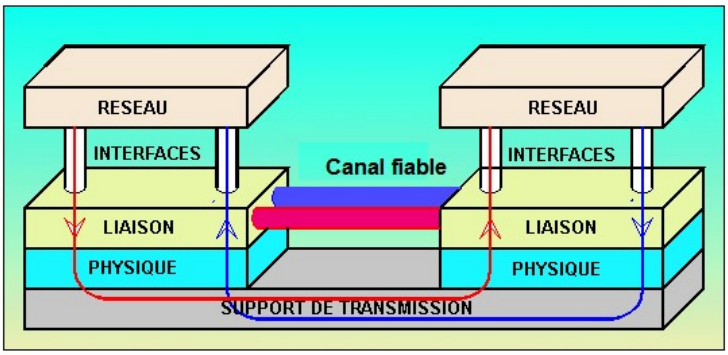


Figure 62. Rôle de la couche "Liaison"

Ce chapitre est divisé en deux parties principales. Dans la première partie, nous allons voir les principales fonctionnalités assurées par la couche liaison, à savoir :

- La structuration en trame : le découpage de flot de bits en trames.
- Le contrôle d'erreurs :
    - Les codes de parité : VRC et LRC.
    - Les codes polynomiaux : CRC

- Les stratégies de retransmission de trames :
    - Numérotation de trames.
    - Acquittements.
    - Protocole “Envoyer et attendre”
    - Protocole “Go-back-N”

Ces fonctionnalités sont implémentées par un ensemble de procédure formant un protocole de la couche liaison. Il est essentiel de comprendre comment ces fonctionnalités sont implémentées par un protocole, pour cela, nous étudions, dans la deuxième partie de ce chapitre, un protocole de référence de la couche liaison, et dont tous les autres protocoles de cette couche sont ses dérivées, à savoir le protocole HDLC.

Partie 1. Principales fonctionnalités de la couche liaison

1 Découpage en trames

Au niveau physique, les données sont une suite de bits, tandis qu’en niveau liaison elles doivent avoir une structure. C’est la couche liaison qui structure et découpe les données en des blocs appelés *trames*. Différentes méthodes de découpage existent, parmi lesquelles :

- *Le découpage orienté caractère* : chaque trame contient une suite de caractères encadrée par un texte de début (STX – Start Text) et un texte de fin (ETX – End Text). Le rôle de STX et ETX est de délimiter la trame pour pouvoir la reconnaitre par le récepteur.
  L’inconvénient majeur de cette méthode est qu’elle dépend du codage des caractères (sur 7 ou 8 bits) qui peut être différent d’une machine à l’autre, et ainsi en utilisant ce découpage il faut penser à l’adapter au codage utilisé par la machine.
- *Le découpage orienté bit à l’aide d’un fanion :* cette méthode consiste à utiliser une suite bien définie pour délimiter un bloc de donnée de taille quelconque. Elle :
    - Permet à une trame d’avoir une longueur quelconque (pas forcément un multiple de caractère).
    - Peut être utilisée quel que soit le codage de caractères utilisé.

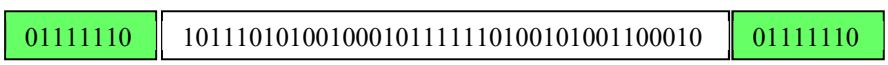
  Cette méthode est celle utilisée en pratique pour le découpage de données au niveau liaison.

1.1 *Découpage orienté bit*

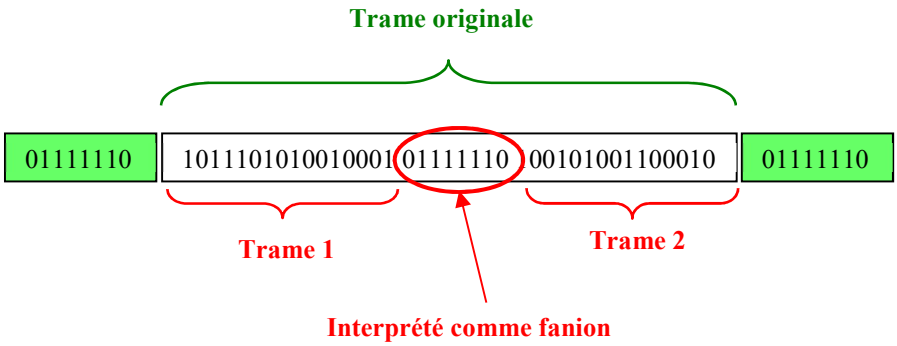
Le principe de cette méthode est d’utiliser un *fanion* (appelé aussi flag ou drapeau), qui est une suite de bits spéciale « 01111110 », dont le rôle est de délimiter la trame, c'est-à-dire permettre de connaitre le début et la fin de la trame.

Exemple : pour la suite de données suivante (provenant de la couche réseau) :

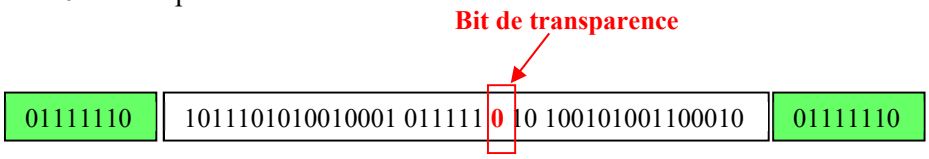
10111010100100010111111010010100110010, la suite à envoyer est :



Le problème qu'on peut rencontrer avec cette méthode est l'*ambigüité* dans la reconnaissance des trames. Ceci peut arriver si les données contiennent eux même la suite fanion, c'est-à-dire « 01111110 », ce qui est le cas pour la suite de données de l'exemple précédent, et ainsi le récepteur va l'interpréter ainsi :



Pour remédier à ce problème on utilise des *bits de transparence* (*Bit stuffing*) : Après 5 bits consécutifs à 1, ajouter un bit 0 de transparence.



Ainsi, à la réception, la couche liaison procède en premier lieu à la suppression des bites de transparence, ensuite elle extrait les trames délimitées par les fanions.

2 Contrôle d'erreurs

Le signal peut subir des altérations au cours de sa propagation dans le média de transport. Ceci à cause de plusieurs facteurs appelés des phénomènes perturbateurs, entre autres le bruit, l'interférence, etc. (pour plus de détails, reportez-vous au chapitre 4). Ces altérations et modifications du signal engendrent des erreurs dans les données transmises. Ces erreurs sont généralement des *erreurs bit*.

Chaque liaison est caractérisée par son *taux d'erreurs* T_e (appelé *BER -Bit Error Rate*), qui dépend surtout du média de transmission qui assure cette liaison (c'est une caractéristique du média de transmission). Le taux d'erreur bit es calculé comme suit :

$$T_e = \frac{Nombre\ de\ bits\ erronés}{Nombre\ de\ bits\ total}$$

Exemple : soit la donnée transmise est : 1110001100111100. Si la donnée reçue est : 1110**1**011001**0**1100 → On dit que le taux d'erreur de cette liaison est : $T_e = \frac{2}{16} = 12{,}5\%$

Exemples de taux d'erreurs : T_e de la fibre optique : ~ 10^{-12}, le canal radio : ~ 10^{-5}.

Pour remédier au problème d'erreurs et avoir une *transmission fiable* (sans erreurs), la couche liaison assure deux services :

- *La détection d'erreurs (obligatoire)* : en utilisant des codes détecteurs.

- *La correction d'erreurs* : en plus de la détection des erreurs la couche assure aussi leur correction. La correction peut se faire par deux méthodes différentes :
    - *Automatique* : en utilisant un code correcteur (détecteur et correcteur), par exemple : le code de Hamming.
    - *Par retransmission* : en utilisant une stratégie de retransmission de trames erronées (en plus d'un code détecteur).

Dans ce qui suit, nous allons voir les différentes techniques de detection, tandis que pour la correction d'erreurs nous allons voir seulement les techniques de correction par retransmission.

3 Techniques de détection d'erreurs

Il existe différentes techniques de détection d'erreurs, certaines d'entre elles ne sont pas applicables à cause des faiblesses qu'elles présentent, tandis que d'autres sont celles utilisées actuellement. Ces méthodes sont :

- Détection par écho.
- Détection par répétition.
- Détection par code :
    - Code simple : les parités.
    - Code polynomial ou code cyclique : CRC - Cyclic Redundancy Check

N.B. La technique de détection utilisée est la détection par code polynomial.

3.1 La détection par écho

Cette méthode consiste à envoyer une trame et son duplicata au récepteur qui va les comparer et vérifier s'il y a une erreur ou non. L'inconvénient de cette méthode est que dans le cas où la même erreur est survenue sur les deux trames, le récepteur acceptera la trame alors qu'elle est erronée.

3.2 La détection par répétition

Le récepteur renvoie la trame reçue à l'émetteur pour vérification d'erreurs. L'inconvénient de cette méthode est que dans le cas où seule la trame renvoyée par le récepteur est erronée, l'émetteur considèrera la trame erronée alors qu'elle est bien reçue par le récepteur.

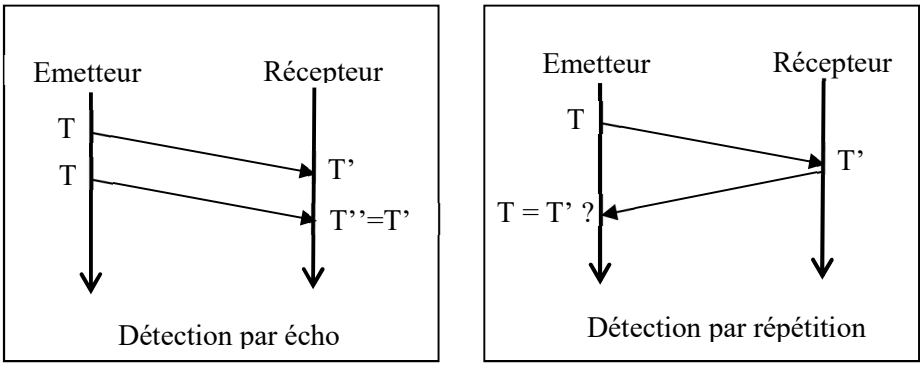


Figure 63. Détection d'erreurs par écho et par répétition

3.3 *La détection par code*

La détection par code (ou par redondance) consiste à ajouter à l'information initiale (*bits de données*) une information supplémentaire appelés *bits de contrôle* (ou bits de redondance). Les bits de contrôle permettent d'insérer à l'information initiale une caractéristique qui sera vérifiée à la réception. Ceci est le principe de fonctionnement des codes détecteurs aussi bien que des codes correcteurs.

Les deux types de codes détecteurs les plus répandus sont :

- *Code simple (de parité) :*
  - VRC (Vertical Redundancy Check).
  - VRC + LRC (Longitudinal Redundancy Check).
- *Code polynomial CRC* (Cyclic Redundancy Code).

3.3.1 Codes simples. La parité verticale (VRC)

La *détection par parité* est le *plus simple* code *détecteur* d'erreurs. Elle consiste à ajouter un seul bit de parité à la fin d'un bloc de données (généralement un caractère). Selon la parité, on a :

- *Parité paire :* Le nombre total de bits à 1 (y compris le bit de parité) doit être *pair*.
  Exemple : 1011011 → 1011011**1**, 0100100 → 0100100**0**
- *Parité impaire :* Le nombre total de bits à 1 (y compris la parité) doit être est impair.
  Exemple : 1101110 → 1101110**0**, 0011000 → 0011000**1**

Faiblesse. Le code de parité permet de détecter la moitié des erreurs. Les erreurs sur un nombre impair de bits peuvent être détectées, ce qui n'est pas le cas pour les erreurs sur un nombre pair de bit. Alors qu'en pratique un nombre pair d'erreurs est aussi probable qu'un nombre impair.

Exemple : message original : 1011011 → message envoyé : 1011011**1**.

- Si le message reçu est : 1**1**110111 → possible de détecter l'erreur (parité est incorrecte).
- Si le message reçu est : 1**1****0**0111 → impossible de détecter l'erreur.

Pour remédier à ce problème, une version plus développée a été proposée, à savoir la détection par parité double.

3.3.2 Codes simples. La parité verticale et horizontale (VRC+LRC)

Ce code assure le contrôle de parité sur un bloc de données qui peut contenir un ensemble de caractères. Dans ce code, la parité verticale (VRC) est complétée par une parité horizontale (LRC) en procédant ainsi :

1) Organiser la séquence de bits sous forme d'une matrice.
2) Calculer une parité par ligne et par colonne.
3) La suite de données à transmettre est :

caractère	VRC	caractère	VRC	……	LRC	VRC

Exemple : Soit à transmettre le mot HELLO (parité paire). En appliquant un code de détection par parité double on aura :

La suite à transmettre est : 10010000100010111001100110011001100111111000010**0**.

La parité VRC + LRC permet de :

- *Détecter* toutes les *erreurs multiples sur 2 ou 3 bits.*
- *Corriger* toutes les erreurs sur *un seul bit* (c'est un *code correcteur* simple).

Faiblesse. A partir de 4 bits erronés, le code peut échouer à détecter les erreurs. Exemple : considérons le mot HELLO de l'exemple précédent, l'altération des quatre bits suivants (voir tableau ci-dessous) génère une erreur non détectable par ce code.

	H	E	L	L	O	LRC
Bit7	1	1	1	1	1	1
Bit6	0	0	0	0	0	0
Bit5	0	1	0	1	0	0
Bit4	1	0	1	1	1	0
Bit3	0	0	1	0	1	0
Bit2	0	0	0	0	1	1
Bit1	0	1	0	0	1	0
VRC	0	1	1	1	1	0

3.3.3 Code polynomial – CRC

Un bloc de *m* bits est représenté par un *polynôme M(x)* de degré *m-1* avec des coefficients *x* binaires (prendre l'une des valeurs 0 ou 1). Exemple : 10110 → $M(x) = x^4+x^2+x^1$.

3.3.3.1 Principe du code CRC

- L'émetteur et le récepteur utilisent un polynôme commun appelé *polynôme générateur G(x)* de degré *r*.
- Le bloc de données à envoyer est transformé *à l'émission* en un *mot de code* polynomial, représenté par un polynôme qui est un *multiple du polynôme générateur G(x).*
- Le récepteur vérifie si le mot reçu est un mot du polynôme générateur (un multiple de G(x)) ou non.

Les opérations à effectuées sont les suivantes :

- ***A l'émission (Calcul du mot de code)***
  - *Étape 1. Multiplication : $x^r * M(x)$.*

    La première étape consiste à ajouter *r* zéros après le bit de poids faible du bloc représenté par M(x). Il contient ainsi (*m* + r) bits, correspondant au polynôme $x^r M(x)$.
  - *Étape 2. Calcul du reste : $R(x) = Reste (x^r * M(x) / G(x))$.*

    On diviser modulo 2 le polynôme $x^r M(x)$ par *G(x)*et on prend le reste de la division *R(x).*
  - *Étape 3. Soustraction : $M'(x) = x^r * M(x) – R(x)$.*

    On fait la soustraire modulo 2 du reste de la division de la suite de bits correspondant au polynôme $x^r M(x)$. Ceci revient à ajouter les *r* bits du reste à la fin de M.

Le résultat de ces opérations est une suite de bits multiple du polynôme *G(x)* (dont le reste de la division sur *G(x)* est nul). On dit que cette nouvelle suite est un mot de code.

- ***A la réception : (Détection d'erreur)***
  - Le récepteur calcule M'*(x) / G(x).*

- Il accepte la trame comme correcte si le reste est 0. Sinon, si le reste de la division n'est pas nul, une erreur est détectée.

N.B. La division modulo 2 est basée sur la soustraction modulo 2. Pour la soustraction modulo 2, on fait un *OU EXCLUSIF*. *Exemple :* 0011 – 0101 = 0110.

3.3.3.2 Exemple Applicatif.

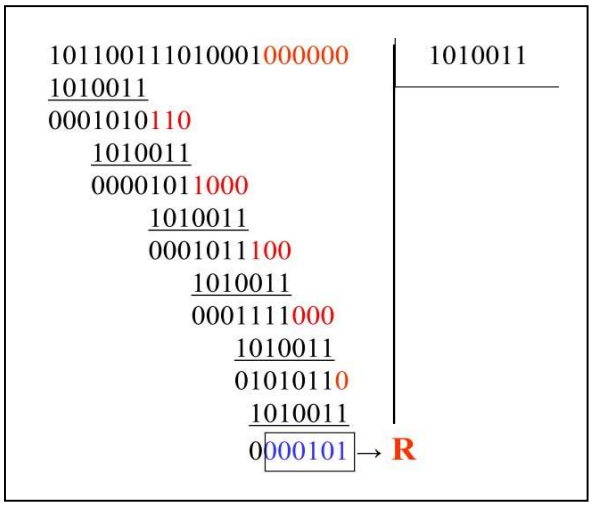
Soit à transmettre la suite M =101100111010001. Le polynôme générateur est $G(x) = x^6 + x^4 + x + 1$.

1) Quelle est la suite réellement transmise ?
2) La suite binaire reçue est : M'=1011001110010100. Est-ce que la transmission est sans erreur ?

Réponse :

1) Le polynôme générateur est : $G(x) = x^6 + x^4 + x + 1 \rightarrow G = 1010011$

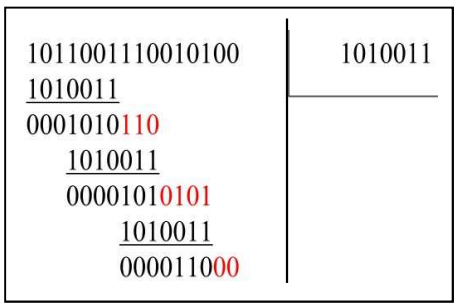
- ***A l'émission :***
  - *Etape 1. Multiplication :* on ajoute r = 6 bits à droite de la suite → 101100111010001**000000.**
  - *Etape 2. Division* : on calcule le reste de la division *R* (sur *r* = 6 bits)



  - *Etape 3. Soustraction :* on ajouter R à la suite de l'étape 1, ainsi la suite à transmettre est : M'= 101100111010001**000101**.

- ***A la réception :***

  La suite binaire reçue est : M'=1011001110010100. On divise la suite par G(x) :



  Le reste R = 11000 ≠ 0, donc la trame reçue est erronée.

N.B. En cas d'une transmission sans erreurs, le message original est obtenu par le récepteur en enlevant les *r* bits situés à droite de la suite reçue.

3.3.3.3 Avantages du code CRC

- Il permet de contrôler des mots d'une longueur variable.
- Une réalisation simple et efficace en matériel.
- Une très bonne capacité de détection d'erreurs.

La capacité de détection d'erreurs du CRC dépond largement du polynôme générateur utilisé. Les polynômes générateurs les plus utilisés sont :

- CRC –12 : $x^{12} + x^{11} + x^3 + x^2 + 1$
- CRC –16 : $x^{16} + x^{15} + x^2 + 1$
- CRC–CCITT: $x^{16} + x^{12} + x^5 + 1$
- CRC-32 : $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$

4 Critères d'efficacité d'un contrôle d'erreurs

L'efficacité d'un contrôle d'erreurs est mesurée par :

- La capacité de détecter et/ou corriger des erreurs multiples.
- La capacité de détecter et/ou corriger des rafales d'erreurs d'une certaine longueur.
- La probabilité d'accepter une trame erronée comme correcte.
- *Le rendement* : le rapport entre les bits de données et la longueur totale des paquets.

$$\boldsymbol{\rho = \frac{m}{(m + r)}}$$

Exemple: Sur un mot de données de 8 bits, si on ajoute 4 bits de contrôle, donc $\rho = 67\%$.

N.B. Ce rendement est calculé dans le cas d'un contrôle d'erreur par redondance.

Important. Toute donnée ajoutée aux données d'utilisateur pour assurer une fonctionnalité quelle qu'elle soit (ex. les bits ajoutés pour la détection d'erreurs) est appelée donnée de contrôle, et elle ralenti la communication. On peut la considérer comme le coût à payer pour réaliser la fonctionnalité désirée. Donc, plus il y a de données de contrôle à ajouter plus l'impact (négatif) sur la communication est important et inversement. Ainsi, on cherche toujours à assurer la fonctionnalité mais avec le moindre coût, tout en minimisant les données de contrôle à ajouter. Notez que les données de contrôle sont les données ajoutées par chaque couche connues sous le nom de PCI (ex. les @MAC au niveau liaison, les @IP au niveau réseau, etc.)

Une fonctionnalité peut être assurée par l'ajout de données ou par la mise en œuvre d'un traitement spécifique (ex. la retransmission de données en cas d'erreur, etc.). De même, tout traitement supplémentaire réalisé pour assurer une fonctionnalité ralenti la communication, et on cherche généralement le bon compromis qui garantit que le service soit bien réalisé avec un coût minimal.

5 Stratégies de retransmission ou ARQ (Automatic Repeat reQuest)

La retransmission des trames erronées est la méthode principale pour la correction des trames en cas d'erreurs de transmission. Elle est plus efficace que les codes correcteurs étant donné que :

- Les codes correcteurs (comme par exemple le code de Hamming) ajoutent beaucoup de données de redondance ce qui influe négativement sur le rendement.
- La vitesse de transmission des médias de transport est très grande, et ainsi on préfère la retransmission de la trame erronée et on perd en temps de propagation mieux que d'ajouter une information redondante qui minimise le rendement du réseau.

Par la suite, on va présenter les quatre stratégies ARQ suivantes :

1. Send & Wait ARQ, « Envoyer et attendre » ou « Stop-and-Go ».
2. Continuous ARQ (Fenêtre glissante) :
    - Le protocole « Go-back-n » (à rejet automatique).
    - Le protocole à « Rejet sélectif ».

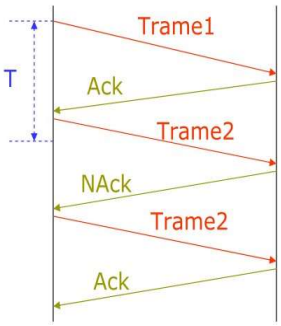
5.1 *Send & Wait (Envoyer et attendre)*

5.1.1 Principe

- S'assurer qu'une trame a été bien reçue avant d'envoyer la suivante, c'est-à-dire l'émetteur envoie une seule trame à la fois et attendre son acquittement.

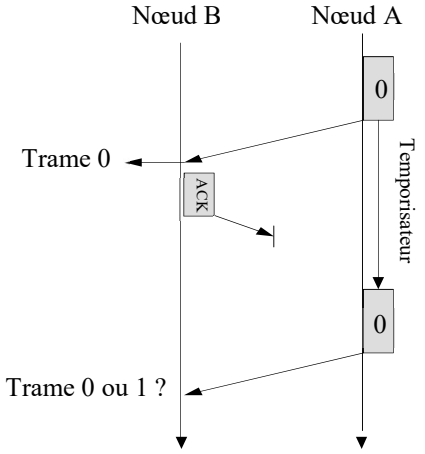
5.1.2 Fonctionnement du Send & Wait

- L'émetteur envoie une trame et attends son acquittement.
- A l'envoi de la trame, une horloge est armée au niveau de l'émetteur pour calculer un *temps de temporisation T (Time out)*, au bout duquel l'émetteur procède à la réémission de la même trame si aucune réponse n'est parvenue du récepteur.
- La trame contient un CRC qui permet de détecter des erreurs.
- Si la trame est correcte, le récepteur renvoie un acquittement positif (ACK).
- Si la trame et erronée, le récepteur renvoie un acquittement négatif (NCAK).
    - ACK et NACK sont également protégés par un CRC.
- Le temporisateur (horloge) est armé à chaque envoi et désarmé à la réception d'un ACK.
- La retransmission peut être :
    - *Implicite* : après expiration du time out.
    - *Explicite* : à la réception d'un acquittement négatif « NACK »



Donc en résumé, on a quatre cas possibles :

- *Cas 1* : La transmission est sans erreurs → Retour d'un acquittement positif (ACK).
- Cas 2 : La transmission est avec erreurs → Retour d'un acquittement négatif (NACK).
- Cas 3 : Une perte de la trame → Retransmission après l'expiration du time out T.
- Cas 4 : Une perte de l'acquittement :
  - Si ACK positif → *Problème :* Duplication de la trame au niveau du récepteur.



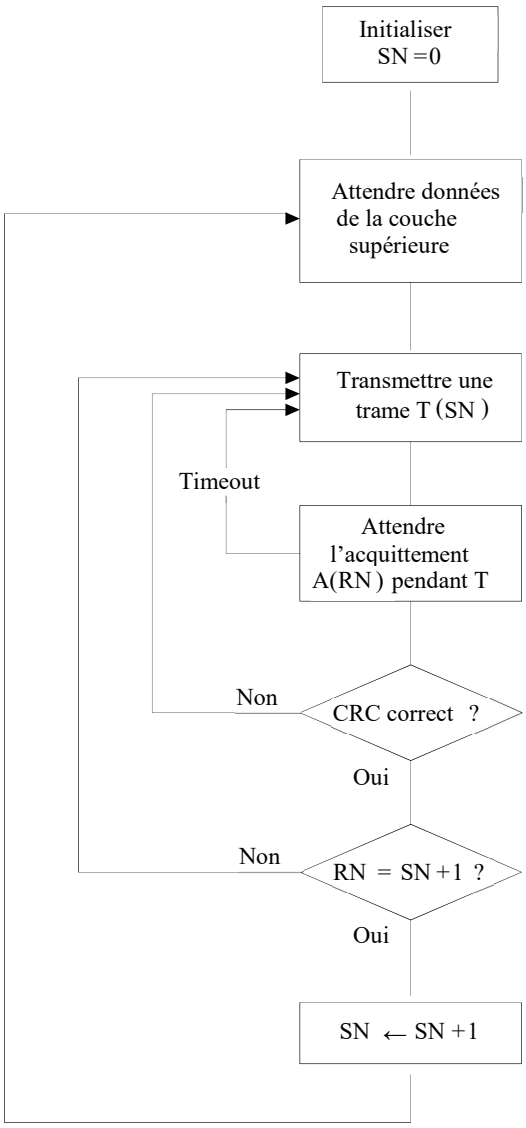
La solution est donc de numéroter les trames.

5.1.3 Numéros de séquence de trames

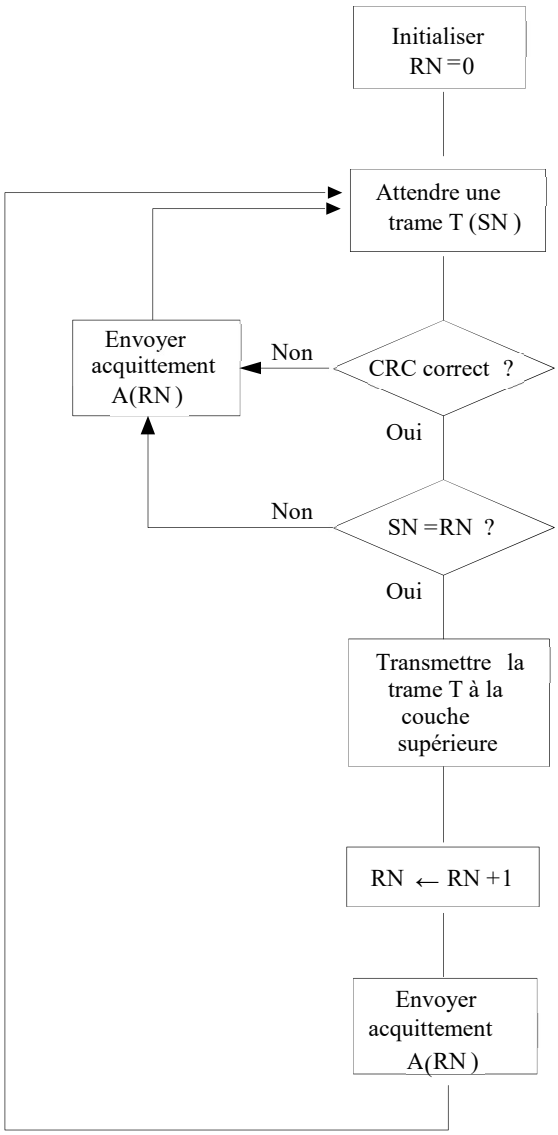
En utilisant la numérotation des trames, chaque trame est identifiée d'une façon unique par un *numéro de séquence SN (Sequence Number).* Ce numéro de séquence est un champ dans la trame (voir la partie 2 de ce chapitre), et étant donné que la trame a une taille finie (un bloc de données limité par des fanions), le SN, lui aussi, est d'une taille limitée. Donc, la question est : de quelle longueur ce SN ?
Dans le cas de Send & Wait, une numérotation sur 1 bit est suffisante (numérotation en alternance 0 ou 1), car selon ce protocole l'émetteur envoie une seule trame à la fois et attend son acquittement.

En appliquant la même idée sur les acquittements, chaque ACK peut contenir un *numéro de séquence RN (Reception Number)* du prochain paquet attendu. Aussi, dans ce cas une numérotation de séquence sur 1 bit est suffisante. Le NACK n'est donc plus nécessaire, puisque le récepteur peut redemander la transmission de la trame précédente avec un ACK numéroté.

Avec la numérotation des trames et des ACK, les algorithmes de l'émetteur et du récepteur selon le protocole Send & Wait sont les suivants :



a) Algorithme de l'émetteur



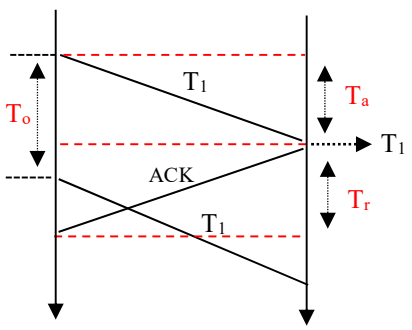
b) Algorithme du récepteur

N.B: Le protocole Send & Wait est adapté à des liaisons Half duplex.

5.1.4 Problème d'interblocage

Le time out doit être bien calculé en fonction du temps d'aller/retour entre l'émetteur et le récepteur pour éviter le problème d'interblocage. Ce problème intervient lorsque le time out T_o est inférieure au temps d'aller d'une trame de l'émetteur au récepteur T_a plus le temps de retour T_r de son ACK :

Si ($T_a + T_r > T_o$) alors il y a interblocage.



5.1.5 *Performances du protocole Send & Wait*

- *Efficacité du protocole (ρ) :*

  L'*efficacité* (le *rendement* ou le *taux d'utilisation*) du réseau est le rapport entre la valeur réelle (effective) d'une grandeur (ex. débit, quantité de données, etc.) et sa valeur théorique. Si on utilise comme grandeur la quantité de données, donc l'efficacité est le rapport entre la quantité de données transmise réellement (c'est la taille d'une trame) sur la quantité de données qui peut être transmise dans le temps écoulé entre l'émission de la trame et la réception de l'acquittement :

$$\boldsymbol{\rho = \frac{Q_{data}}{RTT \cdot D}}$$

Où :

- Q_{data}: *la taille de la trame.*
- RTT: *le temps d'aller/retour*
- D: *le débit*

Le temps d'aller/retour RTT (ou le temps de transfert) est calculé ainsi :

$$RTT = T_t(trame) + T_t(ACK) + 2T_p$$

Où :

- $T_t(trame)$: *le temps de transmission de la trame.*
- $T_t(ACK)$: *le temps de transmission de l'ACK.*
- T_p: *le temps de propagation.*

$$\boldsymbol{RTT = \frac{Q_{data}}{D} + \frac{Q_{ACK}}{D} + 2T_p}$$

Donc, l'efficacité est :

$$\boldsymbol{\rho = \frac{Q_{data}}{Q_{data} + Q_{ACK} + 2DT_p}}$$

N.B. $(RTT \cdot D)$ est appelé « produit largeur de bande – délai » noté *BWD* (*BandWidth Delay product*).

5.2 *Stratégies de retransmission continues (Continus ARQ)*

Selon le protocole « Send & Wait », l'émetteur (resp. récepteur) passe son temps à attendre l'acquittement du récepteur (resp. La trame de données de l'émetteur). On peut procéder par *anticipation*, car l'attente de l'acquittement n'empêche pas la transmission des trames suivantes. Donc, on continue à envoyer des trames pendant qu'on attend l'acquittement de la première trame envoyée.

5.2.1 *Principe*

Le principe des techniques de retransmission continues (ARQ) est de transmettre plusieurs trames avant d'attendre leurs acquittements. Il existe deux types de protocoles ARQ :

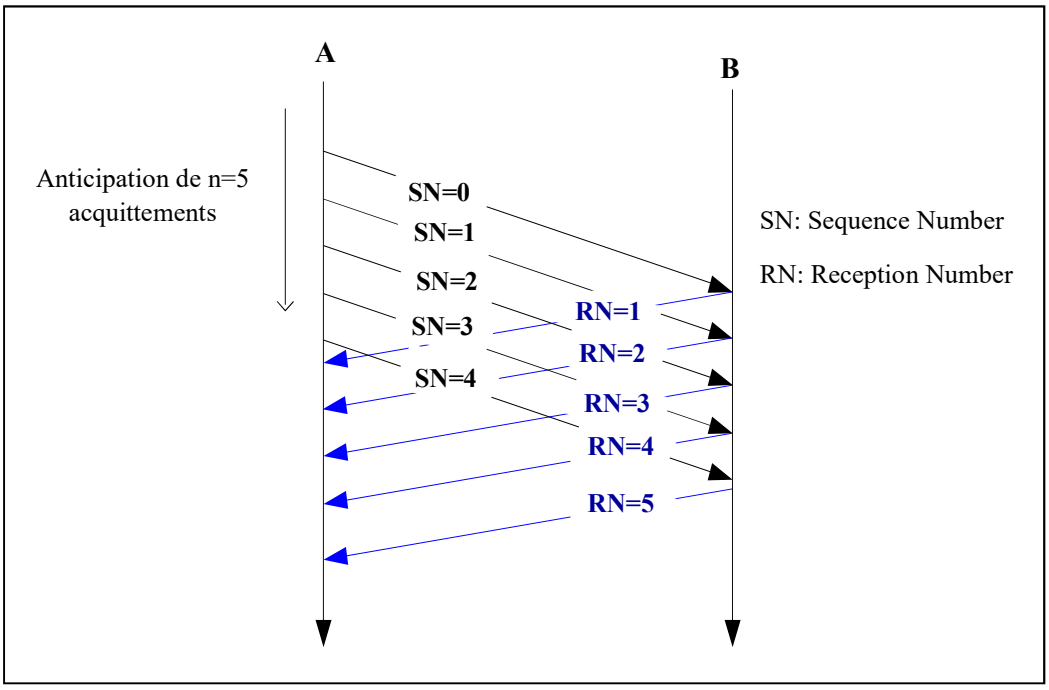
- Go-back-n (Rejet systématique).
- Rejet sélectif.

5.2.2 Protocole « Go-back-n »

C'est la méthode ARQ la plus utilisée au par la couche liaison et transport. Elle fonctionne ainsi :

- L'émetteur envoie plusieurs trames de suite sans attendre d'acquittement.
- Chaque trame est identifiée par un numéro de séquence SN.
- L'émetteur sauvegarde une copie de chaque trame envoyée et non pas encore acquittée afin de pouvoir la retransmettre en cas d'acquittement négatif de la part du récepteur.
- Les trames envoyées et non acquittées sont stockées dans un tampon mémoire appelé *fenêtre coulissante*, *glissante* ou d'*anticipation*.
- Le paramètre *n* dans « Go-back-n » indique la taille du tampon mémoire, c'est à dire le nombre de trames qui peuvent être envoyées sans recevoir d'acquittement.
- Le récepteur *n'accepte que la prochaine trame attendue*, et envoie un acquittement pour chaque trame correctement reçue.

Example: Go-back-5



N.B. $RN = x$ signifie que l'émetteur acquitte les trames $SN < x$ et attend la trame $SN = x$.

5.2.2.1 Fenêtre coulissante - glissante (Sliding Window)

La fenêtre glissante (*Sliding Window*) est un tampon mémoire contenant les *n* trames après la dernière trame acquittée. Si le numéro de la dernière trame acquittée est *x* donc les trames contenues dans la fenêtre ont un numéro qui vérifie :

$$x + 1 < SN < x + n + 1$$

La numérotation de trames permet de détecter :

- L'absence (ou la perte) de trames.
- La duplication de trames (ce qui permet la destruction du duplicata).

- L'ordonnancement des trames (et donc ré-ordonnancement ou destruction des trames n'arrivant pas dans l'ordre).

Après l'acquittement d'une trame transmise, la fenêtre glisse à droite et autorise la transmission d'une nouvelle trame d'où l'appellation de « fenêtre glissante ».

Exemple : soit la fenêtre coulissante de taille 9 (voir *Figure 64*). Le glissement est montré par des pointillés.

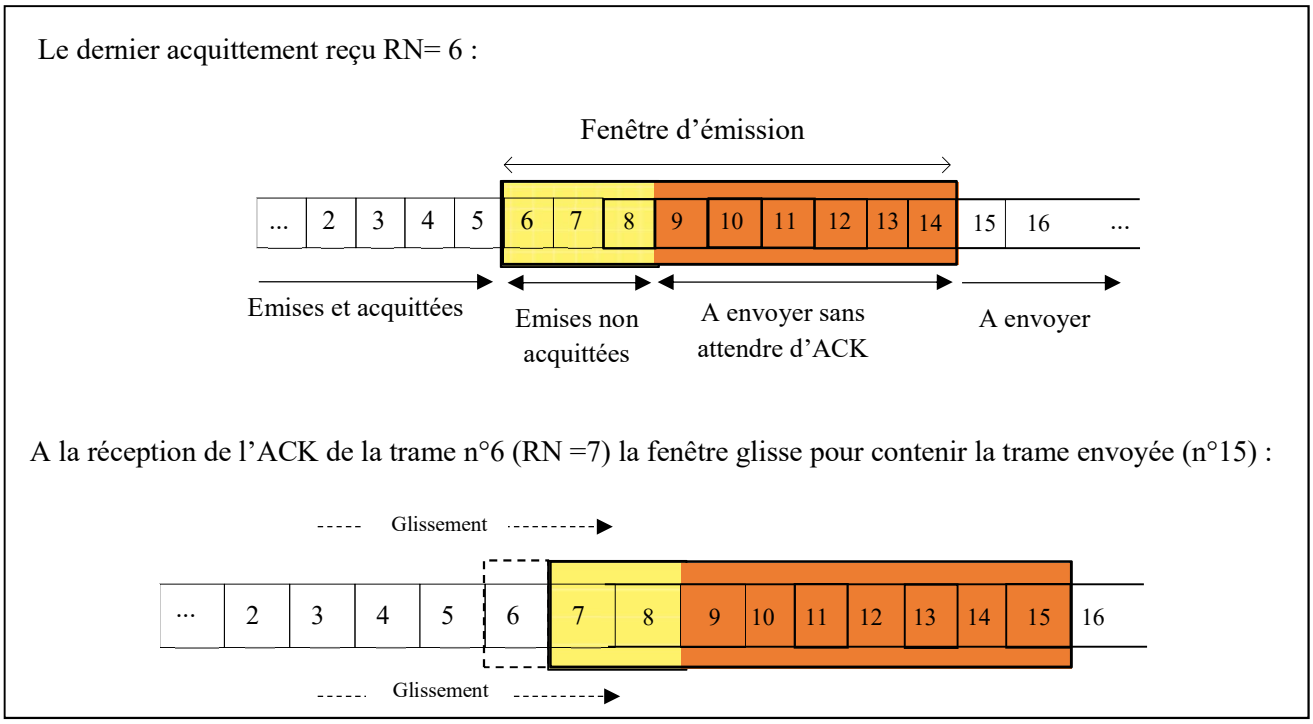


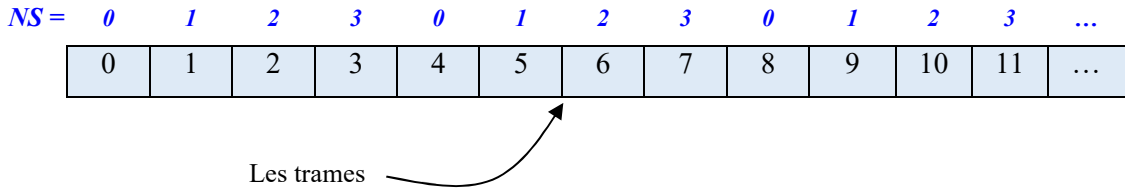
Figure 64. Fonctionnement d'une fenêtre coulissante de taille n = 9.

5.2.2.2 Numéros de séquence des trames

Le numéro de séquence de la trame est d'une longueur limitée et généralement fixe (car la trame est d'une taille bornée). La numérotation se fait modulo M ou $M = 2^{(taille\ du\ champ\ de\ numérotation)}$.

Car la numérotation est modulo un nombre M donc au cours d'une même connexion un même numéro peut être réutilisé pour identifier différentes trames.

Exemple : supposons qu'on a 12 trames à envoyer, et qu'on fait la numérotation modulo 4. On peut avoir un même numéro de séquence utilisé pour identifier trois trames différentes.



Le fait d'identifier deux trames différentes par le même numéro peut provoquer une ambigüité au niveau du récepteur. Ceci est le cas où le récepteur acquitte la réception positive de la première trame (dont le numéro de séquence $SN = 0$). Ainsi il envoie un acquittement positif en demandant la transmission de la quatrième trame. Si l'acquittement est perdu, l'émetteur va renvoyer la première trame (dont $NS = 0$) après expiration du time

out, qui sera considérée par le récepteur comme étant la quatrième trame car, elle aussi, son numéro de séquence est *NS* = 0. Donc, pour éviter l'ambigüité, il faut que :

$$\boxed{M \geq n + 1}$$

n : Taille de la fenêtre d'anticipation,

M : capacité de numérotation

5.2.2.3 Taille optimale de la fenêtre glissante

La taille optimale de la fenêtre est la taille qui permet une transmission efficace, c'est-à-dire une transmission continue sans arrêt. Pour cela, il faut que :

$$n * Q_{data} = RTT * D$$

$$\boxed{n = \frac{RTT * D}{Q_{data}}}$$

N.B. Dans ce cas, la capacité de la liaison de données est totalement utilisée.

5.2.2.4 Avantages et inconvénients du Go-back-n

Selon Go-back-n, la réception d'un NACK en cas d'une trame erronée ou l'expiration de son time out provoquent la retransmission de toutes les trames depuis la trame en question (erronée). Dans ce cas, on parle de ***rejet systématique***. Les deux points principaux caractérisant le rejet systématique sont les suivants :

(+) *Il garder le séquencement des trames.* Ce qui est un avantage de ce protocole car le récepteur est sûr de recevoir les trames dans l'ordre et par conséquent il n'a pas besoin de les réordonner.

(-) *Il élimine des trames bien reçues.* En effet, il y a des trames qui seront retransmises malgré qu'elles sont bien reçues, tout simplement parce qu'elles sont hors séquencement. Ceci est un inconvénient car tout traitement supplémentaire a un impact négatif sur les performances du réseau.

Pour remédier à cela, on peut retransmettre seulement la trame erronée, et on parle dans ce cas de ***Rejet sélectif.***

5.2.3 Rejet sélectif

Dans le protocole « Rejet Sélectif », lorsqu'une trame est perdue (expiration du time out) ou erronée, le récepteur renvoie un acquittement négatif avec le numéro de séquence de cette trame. Dans ce cas, l'émetteur renvoie seule la trame en question (voir *Figure 65*).

5.2.3.1 Inconvénient par rapport à Go-back-n

Les principaux inconvénients du protocole « Rejet Sélectif » par rapport au protocole « Go-back-n » sont les suivants :

- Un nouveau type d'acquittement négatif doit être utilisé pour indiquer un rejet sélectif et le distinguer d'un rejet systématique (par exemple, le protocole HDLC utilise REJ → rejet systématique, SREJ→ rejet sélectif).

- La mémorisation de toutes les trames reçues hors séquencement, et par conséquent la nécessité d'avoir un tampon mémoire (buffer) au niveau du récepteur (ce qui n'est pas le cas pour le rejet systématique). Le tampon mémoire de réception a la même taille que la fenêtre d'émission.
- Le récepteur doit gérer le tampon de réception, et ainsi réordonner les trames retransmises hors séquence (voir *Figure 65*).

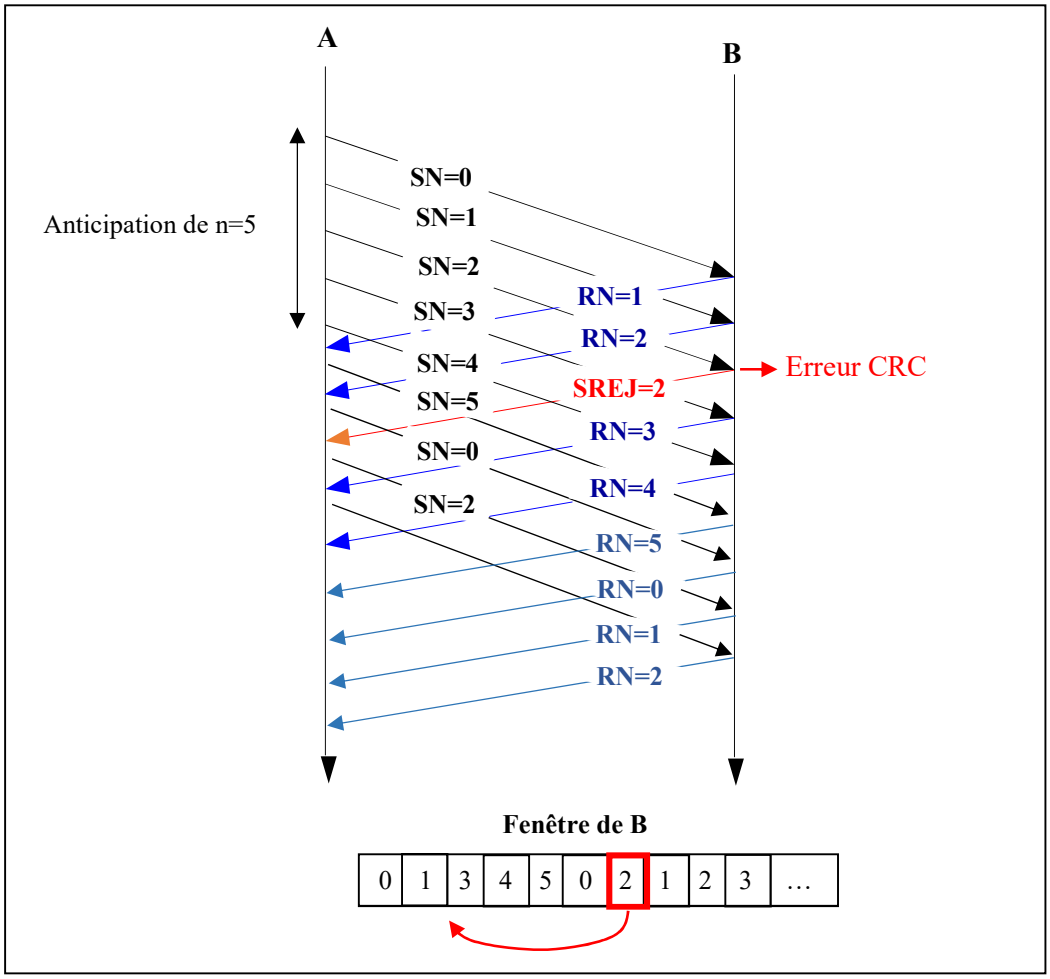


Figure 65. Fonctionnement du "Rejet Sélectif".

La *Figure 65* présente un scénario de fonctionnement selon le « Rejet Sélectif » dans le cas d'une fenêtre coulissante de taille n = 5. On a la trame dont le numéro NS = 2 est erronée (détection par CRC), donc le récepteur demande la retransmission individuelle de cette trame. Après la retransmission, la trame arrive hors séquencement. Par conséquent, le récepteur doit réordonner toutes les trames reçues (c'est pour ça qu'il a besoin lui aussi d'un tampon).

5.2.3.2 Avantage par rapport à Go-back-n

Pour illustrer l'avantage du « Rejet Sélectif » par rapport à « Go-back-n », nous allons comparer la surcharge du réseau dans les deux cas. Notez que la surcharge du réseau dans notre cas se mesure en fonction des données retransmises en cas d'erreurs. Elle est égale à la fraction des trames retransmises dans un nombre de trames données :

$$Surcharge = \frac{Quantité_données_retransmise}{Quantité_données_totale}$$

Exemple. S'il y a une retransmission de 2 trames (erronées) pour chaque transmission de 10 trames maximum, on dit que la surcharge est de 2/10.

(i) Cas d'un rejet systématique.

Les données retransmises en cas d'erreur (dans le pire des cas) = $min(RTT * D, n * Q_{data})$.

Le pire des cas se manifeste quand c'est la première trame qui est erronée. Dans ce cas, c'est toutes les trames de la fenêtre qui seront retransmises. Ainsi, la quantité de données retransmise est égale soit à :

- $RTT * D$, qui est la taille optimale de la fenêtre, et ceci dans le cas d'une transmission efficace (continue).
- La taille de la fenêtre (c.-à-d. $n * Q_{data}$), si elle est inférieure à la quantité de données qui peut être (théoriquement) transmise avant la réception de l'ACK de la première trame.

(ii) Cas d'un rejet sélectif.

Les données retransmises en cas d'erreur est égale à la taille d'une trame Q_{data}, car on fait la retransmission seule de la trame erronée.

Ces deux quantités de données sont retransmise en cas d'erreur dans la transmission d'une quantité totale de $RTT * D$ et donc :

$$(Surcharge)_{Rejet\ séléctif} = \frac{Q_{data}}{RTT * D} \ll (Surcharge)_{Go-back-n} = \frac{n * Q_{data}}{RTT * D}$$

Exemple applicatif. Soit une liaison d'un débit de 1 Mb/s, avec un temps d'aller/retour RTT = 10 ms. Si la probabilité d'erreur paquet est 10^{-4}. Comparer la surcharge du réseau dans les deux cas, d'un Go-back-n et d'un rejet sélectif, pour une taille de trame égale à 10^3 bits.

Réponse.

P (paquet erroné) = 10^{-4} ⇨ Donc, un paquet erroné toute les 10^4 paquets, c.-à-d. toute les $10^4 * 10^3 = 10^7$ bits.

On a : $RTT * D = 10 * 10^{-3} * 10^6 = 10^4 bits$

- Pour le protocole « Go-back-n » (rejet systématique) :
  Surcharge = $10^4 / 10^7 \approx 0.1$ %
- Pour le protocole « Rejet sélectif) :
  Surcharge = 10^3 b/s / $10^7 \approx 0.01$%

5.2.4 Rejet systématique vs rejet sélectif

On applique :

- *Le rejet systématique :* Si le temps de propagation est négligeable (ex. la fibre optique).
- *Le rejet sélectif :* Si le temps de propagation est grand (ex. liaison satellitaire).

5.2.5 Taux d'utilisation des Continus ARQ (Go-back-n et Rejet Sélectif)

Considérons seulement le cas sans erreurs, le taux d'utilisation est le rapport entre la quantité de données transmise réellement sur la quantité de données qui peut être transmise dans le temps écoulé entre l'émission de la trame et la réception de l'acquittement :

$$\boldsymbol{\rho = \frac{n \cdot Q_{data}}{RTT \cdot D}}$$

Où :

- Q_{data}: *la taille de la trame.*
- RTT: *le temps d'aller/retour.*

- Q_{data}: *la taille de la trame.*
- RTT: *le temps d'aller/retour.*

Nous avons déjà vu que :

$$RTT = T_t(trame) + T_t(ACK) + 2T_p$$

$$= \frac{Q_{data}}{D} + \frac{Q_{ACK}}{D} + 2T_p$$

Donc, l'efficacité est :

$$\boldsymbol{\rho = \frac{n \cdot Q_{data}}{Q_{data} + Q_{ACK} + 2DT_p}}$$

N.B. Les protocoles *Go-back-n* et *Rejet sélectif* peuvent atteindre un taux d'utilisation (sans erreurs) $\boldsymbol{\rho = 1}$.

5.2.6 Biggybacking et fenêtre coulissante

Le Biggybacking est le système d'anticipation utilisant la notion d'acquittements groupés. En utilisant le Biggybacking dans un transfert entre deux stations *A* et *B*, un acquittement d'une des deux stations acquitte toutes les trames de l'autre station et dont les numéros sont inférieurs au numéro de cet acquittement. Aussi, les trames d'une station peuvent être acquittées par une trame de données de l'autre station.

Jusqu'ici, nous avons vu les différentes fonctionnalités de la couche liaison (fragmentation de données en trames, contrôle d'erreurs, retransmission des trames erronées ou perdues, etc.). En réalité, ces fonctionnalités sont réalisées et assurées à travers des protocoles. Donc, ces fonctionnalités sont implémentées par un ensemble de procédure formant un protocole de la couche liaison. Plusieurs protocoles existent, un exemple de ces protocoles est HDLC. Afin de comprendre comment les fonctionnalités de la couche liaison sont assurées en pratique, nous allons étudier le protocole HDLC.

Partie 2. Protocoles de la couche liaison

Pourquoi le protocole HDLC ? Tout simplement car c'est le protocole de base (de référence) de la couche liaison, et presque tous les protocoles de cette couche héritent et implémentent ces fonctionnalités.
Partie

1 Le protocole HDLC

Le protocole HDLC est un protocole de référence de niveau 2 (couche liaison) dont l'origine est le protocole SDLC d'IBM. Plusieurs protocoles répandus de cette couche sont des *dérivés de HDLC* comme LAP-B, LAP-D, PPP, etc. (Ils ont tous la même origine, à savoir SDLC). Donc, ces protocoles sont des *variantes* du protocole HDLC, c'est-à-dire ils implémentent un sous-ensembles de ses fonctionnalités.

Le protocole HDLC assure un service de transfert *fiable* et *efficace* entre deux systèmes adjacents, et prend en considération les différents modes de fonctionnement : synchrone / asynchrone, Half-duplex / Full-duplex, liaison point à point / multipoints, etc.

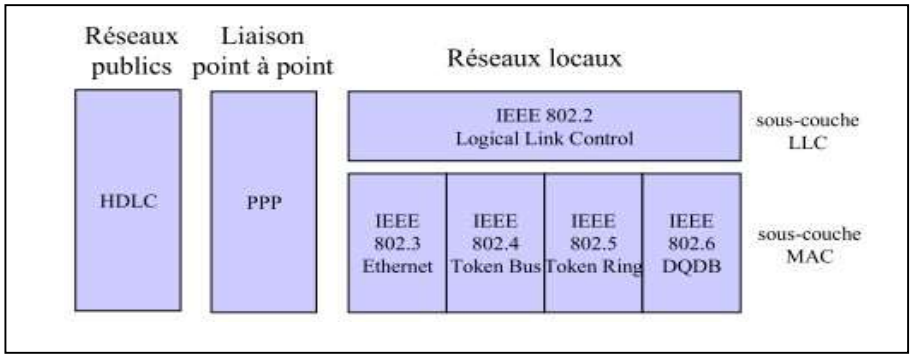


Figure 66. Les protocoles de la couche liaison.

1.1 Les caractéristiques générales du HDLC

Le protocole HDLC est caractérisé par ce qui suit :

- *Un protocole Orienté bit.* Le découpage en trames se fait à l'aide d'un fanion. Donc, la trame HDLC contient un nombre quelconque de bits sans nécessité d'avoir un multiple de 8 bits.
- *Contrôle d'erreur par code acyclique.* Il utilise un code cyclique (CRC) sur 16 bits pour la détection d'erreurs.
- *Utilisation des numéros de séquence.* Le champ de numérotation est codé sur 3 bits (extensible à 7 bits), et ainsi la numérotation de trames se fait modulo M = 8 (ou M = 128).
- *Application des techniques de retransmission.* Il implémente la stratégie de retransmission « Go-back-n » avec fenêtre glissante de 7 (extensible à 127) trames et un rejet systématique (obligatoire) ou sélectif (optionnel).
- *Garantie du Contrôle de flux.* Il implémente la procédure de contrôle de flux (qui est une fonctionnalité de la couche liaison) en des trames d'acquittement (les trames RNR– *Receive Not Ready* et RR– *Receive Ready*).
- *Adaptation au différents types de liaison.* Le protocole HDLC est utilisable sur une liaison *point-à-point* ou *multipoint.*

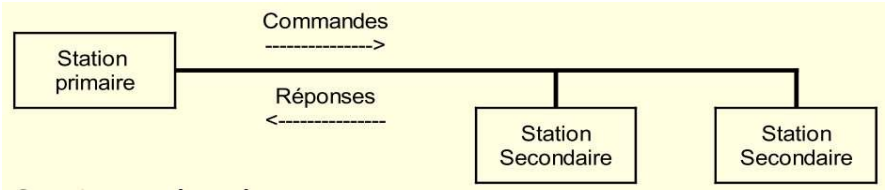
1.2 *Les Modes de fonctionnement*

En générale, il existe deux principaux modes de fonctionnement ; le mode non équilibré et le mode équilibré :

- Le mode Non équilibré - LAP (Link Access Procedure)

  Ce mode de fonctionnement se réalise sur une liaison point à point ou multipoints. Dans ce mode de fonctionnement, il existe deux types de stations :

  - *Station primaire (maître)* : elle est unique, et c'est elle seule qui initialise l'échange de données. Elle envoie des trames de commande.
  - *Stations secondaire (esclave)* : les stations secondaires envoient des trames de réponse, et n'émettent que sur sollicitation de la station primaire.



- *Le mode équilibré (Balanced Link Access Procedure) :*

  Ce mode de fonctionnement se réalise sur des liaisons point à point. Chaque station peut émettre des trames de commande et d'information.

Notez qu'une trame d'information contient des données, tandis qu'une trame de commande (appelée aussi trame de contrôle) est une trame soit d'acquittement (ACK ou NACK), ou une trame de gestion de liaison (ex. trame d'établissement de connexion, libération de connexion, etc.)

Considérons ces deux modes de fonctionnement, le protocole HDLC définit trois modes comme suit :

- *NRM – Normal Response Mode.*
- *ARM – Asynchronous Response Mode.*
- *ABM – Asynchronous Balanced Mode.*

1.2.1 NRM - Mode normal de réponse (Normal Response Mode)

Ce mode est utilisé pour la communication entre une station primaire et une ou plusieurs stations secondaires. Il fonctionne selon le principe Maître-esclave (non équilibré), donc :

- Seule la station primaire est responsable de l'ouverture et de la supervision de la liaison.
- Les stations secondaires :
  - Répondent aux commandes de la station primaire.
  - Elles ne peuvent transmettre que sur invitation.
- Ce mode est utilisable sur des liens point-à-point ou multipoint (il n'est presque plus utilisé).

1.2.2 ARM - Mode asynchrone de réponse (Asynchronous Response Mode)

Ce mode fonctionne selon le principe Maître-esclave, donc il est similaire à NRM. Dans ce mode :

- Les stations secondaires peuvent transmettre librement, sans attendre d'invitation, cependant les transmissions simultanées provoquent une collision.
- Ce mode est rarement utilisé en pratique.

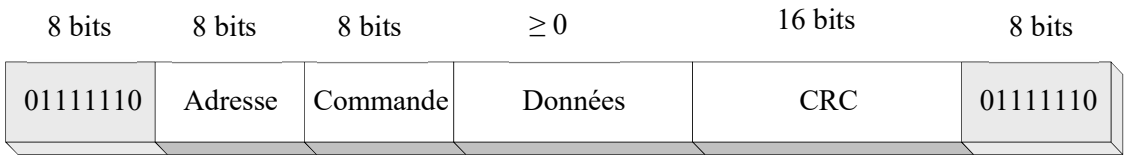
1.2.3 ABM - Mode asynchrone équilibré (Asynchronous Balanced Mode)

Le mode ABM est le mode équilibré du protocole HDLC. Dans ce mode :

- Les stations sont mixtes, donc à la fois primaire et secondaire.
- Chaque station peut initialiser et terminer la liaison (chaque station peut émettre librement).
- Ce mode est principalement utilisé sur les liaisons point-à-point et full-duplex (C'est le mode le plus utilisé).

1.3 *Trame HDLC*

La trame HDLC a la structure suivante :



- *Taille de la trame :*
  - Limitées (pas obligatoirement un multiple d'octet).
  - Peut-être fixe ou variable. Par exemple, la trame HDLC → Taille variable, par contre la trame ATM (qui est une variante du HDLC) → Taille fixe.
- *Fanion (8 bits) :*
  - Son rôle est de délimiter la trame.
  - Une trame peut avoir un fanion de début et un fanion de fin, ou uniquement un fanion de début (qui va jouer les deux rôles, à savoir un délimiteur de début et de fin, pour les trames successives).
  - Pour éviter l'ambigüité (le cas d'avoir la suite du fanion dans les autres champs de la trame), les bits de transparence (bits stuffing) sont utilisés.
  - Après au moins 7 bits consécutifs à 1 → Abandon de la transmission.
  - Une suite de plus de 15 bits consécutifs à 1 → indique une liaison hors service.
- *Adresse (8 bits extensible à 16 bits) :*
  - Il permet d'identifier le destinataire dans une liaison multipoint.
  - Il permet de distinguer « commandes » et « réponses » dans une liaison point-à-point.
- *CRC (16 bits) :*

- Le polynôme générateur du code CRC utilisé est : $G(x)=x^{16}+x^{12}+x^5+1$.

- *Contrôle ou Commande (8 bits extensible à 16 bits) :*
  - C'est le champ le plus important. Il définit le type de la trame, à savoir : trame d'information (I), trame de supervision (S) ou trame non numérotée (U).

1.3.1 Trame d'information (I)

La trame d'information est utilisée pour le transfert de données (Transport des informations). Elle possède le champ « Données ».

Le format du champ « Commande » de la trame d'information est :

Ecriture *:*

I N(S), N(R), P/F

Avec :

- *N(S)* : Numéro de séquence de la trame émise.
- *N(R)* : Numéro de séquence de la trame attendue.
- *P (Poll)* : invitation à émettre dans le cas d'une commande.
- *F(Final)* : fin dans le cas d'une réponse.

1.3.2 Trame de supervision (S)

La trame de supervision est utilisée pour le contrôle d'erreurs (ACK positif → trame correcte, ACK négatif → trame erronée). Elle est utilisée aussi pour le contrôle de flux (le récepteur peut informer l'émetteur s'il est prêt à recevoir des trames ou non).

Le format du champ « Commande » de la trame de supervision est :

Ecriture *:*

I N(R), P/F

Selon le sous-champ « SS », quatre types de trame S existent : *RR, RNR, REJ, SREJ.*

- *SS = 00 → RR (Received & Ready)* : c'est une trame d'acquittement positif.
  - Elle confirme la réception des trames d'information de N° < N(R).
  - Elle demande la transmission des trames suivantes (prêt à recevoir).
- *SS = 10 → RNR (Received & Not Ready) :* c'est une trame d'acquittement positif et de contrôle de flux.
  - Elle confirme la réception des trames d'information de N° < N(R) (ACK positif)
  - Elle demande l'arrêt de la transmission des trames suivantes.
- *SS = 01 → REJ (Reject systématique) :* c'est une trame d'acquittement négatif (contrôle d'erreurs) et de rejet systématique (Go-back-n).
  - Elle confirme la réception des trames d'information de N° < N(R).

- Elle demande la retransmission des trames à partir de N(R) (N°≥ N(R)).

- *SS = 11 → SREJ (Selective Reject) :* c'est une trame d'acquittement négatif et de rejet sélectif.
  - Elle confirme la réception des trames d'information de N° < N(R).
  - Elle demande la retransmission de la trame de N° = N(R.)

1.3.3 Trame non numérotée (U)

La trame non numérotée est utilisée pour la gestion de la liaison : établissement de connexion, libération de la connexion, désignation du mode de fonctionnement ABM, ARM, etc.

Le format du champ « Commande » de la trame non numérotée est :

1bit	1bit	2 bits	1bit	3 bits
1	1	MM	P/F	MMM

- Les bits M permettent de coder 32 types de trames non numérotées. Exemples :
  - *SABM* : Etablissement de la connexion [MMMMM] = [11110].
  - *DISC* : Libération de la connexion [MMMMM] = [11010].
  - *UA* : Confirmation de la connexion [MMMMM] = [00110].
  - *FRMR* : Récupération des erreurs [MMMMM] = [11011].

Le tableau suivant résume les différents cas du champ « commande » :

Types de trame	Champ Commande							
I	0	N(S)			P/F	N(R)		
S	1	0	Type		P/F	N(R)		
U	1	1	M	M	P/F	M	M	M

- *N(S)* : Numéro de séquence de la trame.
- *N(R)* : c'est un acquittement – Indique le prochain numéro de séquence attendu.
- *Type* : Indique le type de la trame (RR, RNR, etc.).

1.3.4 Poll/Final (Invitation à émettre / Fin)

Le bit P/F est utilisé surtout en mode NRM (Maître-esclave). La station primaire le positionne à « P » pour demander une réponse de la station secondaire, qui peut être un acquittement ou une trame d'information (I). Tandis que la station secondaire le met à :

- « P » dans toutes les trames sauf la dernière d'une séquence de trames.
- « F » pour signaler qu'elle termine sa transmission.

1.4 *Contexte de la liaison*

Le contexte de la liaison est l'ensemble des valeurs décrivant la configuration de la liaison et son **:**

- *La configuration de la liaison.* Elle est décrite par l'ensemble de constantes (les mêmes valeurs dans les deux extrémités). Exemple :
  - *T1* : Délai de garde.
  - *N1 :* Taille maximale d'une trame.
  - *W* : Taille de la fenêtre d'anticipation.
  - *T2* : Délai d'acquittement (time out).
  - *N2* : Nombre maximum de rémissions d'une même trame.
- *L'état de la liaison.* Elle est décrite par l'ensemble de variables. Exemple :
  - *V(S)* : Numéro de la prochaine trame I à émettre.
  - *V(R)* : Numéro de la prochaine trame I à recevoir.
  - *DN(R)* : Numéro du dernier acquittement reçu.

  Les valeurs initiales : $V(S) = V(R) = DN(R) = 0$.

1.5 *Cas de rejet de trames*

Selon HDLC, une trame peut être rejetée dans l'un des cinq cas suivants :

- Détection d'erreurs par FCS.
- Taille de la trame < 4 Octet (sans fanion).
- V(R) (local) ≠ N(S) (de la trame).
- Format invalide de la trame (champs d'adresse non valide, champs contrôle non valide, etc.).

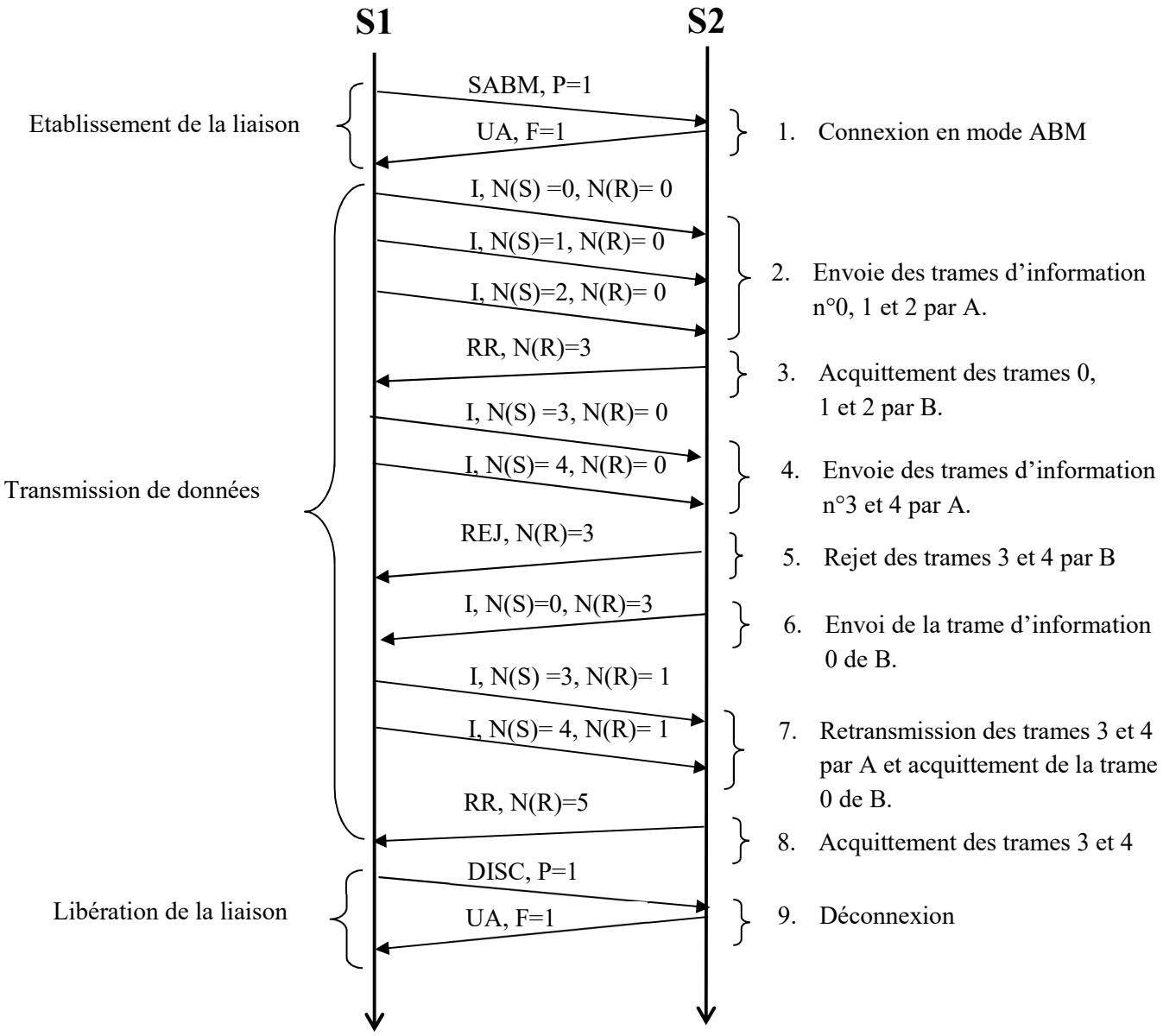
1.6 *Description des échanges*

Selon HDLC, l'échange de données entre deux stations *S1* et *S2* passe par trois étapes :

- *Etape 1. Etablissement de la liaison.*
  - Emission des trames non numérotées SABM et UA.
  - Le bit P/F est positionné à 1 dans la trame de commande SABM invitant l'autre station (soit *S2*) à répondre.
  - La station *S2* émet un acquittement UA avec le bit F à 1.
- *Etape 2. Transmission des données*
  - Transmission avec gestion des numéros de séquence N(S) et N(R).
  - L'acquittement des trames I émises jusqu'au numéro N(S) = x est réalisé par l'envoie d'une trame RR ou I (Biggybacking) avec N(R) = $x+1$ (c.-à-d. le numéro de la trame attendue est $x+1$).
  - La trame REJ avec N(R) = x signifie le rejet des trames N(S) ≥ x.
- *Etape 3. Libération de la liaison (déconnexion)*
  - Emission des trames non numérotées DISC et UA pour terminer l'échange.

Exemple : le schéma ci-dessous décrit le scénario suivant d'échange des données entre deux station S1 et S2 selon HDLC :

a) La station S1 initialise l'échange (1) en demandant à S2 d'établir une connexion en mode équilibré (ABM).
b) S1 envoie les trames d'information n° 0, 1 et 2, qui seront acquittées par la station S2 dans (3).
c) Ensuite S1 envoie les trames d'informations n°3 et 4 dans (4), où la trame n°3 sera rejetée par la station S2 (5) car elle est erronée, et automatiquement les trames qui la suivent (rejet systématique).
d) La station S2 continue à envoyer sa première trame d'information n°0 (6).
e) La station S1 retransmet les trames n°3 et 4 et acquitte la trame n°0 de S2 par Biggybacking.
f) La station S2 acquitte la bonne réception des trames n°3 et 4 de S1.
g) En fin, la station S1 demande la terminaison de la liaison qui va être accordée par la station S2.



2 Conclusion

La couche liaison s'assure de la bonne réception par le destinataire d'une donnée transmise dans le réseau par un équipement source directement connecté. On entend par la bonne réception que le destinataire recevra exactement la même donnée envoyée par la source, et pour cela il faut mettre en place des mécanismes pour détecter tout changement dans la donnée au cours du transfert. Aussi des mécanismes pour se mettre d'accord, l'émetteur et le récepteur, sur comment procéder pour envoyer cette donnée (une par une ou en groupe, etc.) ainsi que comment s'assurer de sa bonne réception (utilisation des ACK). Tous ces mécanismes sont assurés par la couche liaison à travers des protocoles qui fournissent toutes les fonctionnalités nécessaires. Dans ce chapitre, nous avons vu en détail ces mécanismes et ainsi étudié un protocole de référence qui les implémente, et dont presque tous les protocoles de cette couche s'inspirent de lui, à savoir le protocole HDLC.