Introduction

Le but de ce laboratoire était de mieux comprendre l'architecture d'un réseau multipoint en utilisant une station primaire connectée avec deux stations secondaires. Le but était aussi de se familiariser avec le rôle et l'importance de différents protocoles, dont le protocole de la couche de liaison de données, soit le HDLC. Aussi, le but est d'être capables de maitriser le mécanisme de la fenêtre coulissante d'anticipation.

Le protocole High-Level Data Link Control (HDLC) est un protocole qui supporte les communications half-duplex et full-duplex. Il est possible de faire des connexions point-à-point ou multipoints. Au départ, il avait été développé dans le but de faire de la connexion multipoints. Aujourd'hui, par contre, il est surtout utilisé pour faire la connexion directe entre deux dispositifs.

Il y a trois types de stations lorsqu'un système doit utiliser le protocole HDLC :

1. Les stations primaires servent à :
   - faire le contrôle des opérations de liaison
   - s'assurer que les trames issues sont définies comme des commandes
   - faire le maintient d'une liaison logique distincte avec chaque station secondaire.

2. Les stations secondaires :
   - Elles sont sous le contrôle d'une station primaire avec laquelle chacun on la permission de parler un à la fois.
   - Les trames envoyées de ces stations sont considérées comme des réponses pour la station primaire.

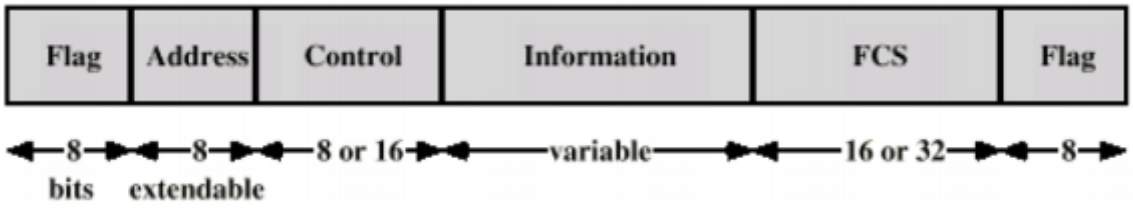
3. Les stations combinées :
   - Elles peuvent agirent comme station primaire et secondaire. Elles peuvent donc fournir des commandes ou des réponses.

Il y a trois modes de transmissions possibles avec ce protocole :

- Mode normal de réponse (NRM)
- Mode balancé synchrone (ABM)
- Mode de réponse asynchrone (ARM)

Il y a deux types de configuration pour le protocole HDLC, soit non-balance, soit balancée. Pour une configuration non balancée, il faut une station primaire et au moins une station secondaire liée à celle-ci. Pour une configuration balancée, il faut deux stations de type combiné.
Pour ce laboratoire, le mode normal de réponse (NRM) est utilisé, qui est un mode non balancé, puisqu'il y a une station primaire et des stations secondaires. Pour fonctionner selon ce mode, il faut que la station primaire initie le transfert vers la station secondaire. Cette dernière pourra ainsi envoyer une réponse, car elle peut seulement transmettre des données en tant que réponse à une commande envoyée par la station primaire.

Dans ce lab, nous utilisons le mode normal de réponse (NRM), ce qui est un mode non-balancé.



Chaque trame est constituée d'un délimiteur de trame, de l'adresse et du contrôle Il contient aussi l'information et le FCS. Le délimiteur de trame, ou bien les flags, a une taille de 8 bits chaque et sert à identifier où la trame commence et finit. En NRM, l'adresse est de 8 bits mais peut contenir plus, est celle du terminal secondaire. Le contrôle sert à déterminer le type de la trame. Dans ce lab, nous utilisons un contrôle de 8 bits.
Les trames peuvent être du type I, S ou U

- Le type I, ou bien information, transporte les données de l'usager de la couche de réseau.
- Le type S, ou bien supervision, est utilisé pour le contrôle de débit et d'erreur. Elle ne contient pas de champ d'information.
- Type U, ou bien non numéroté, est utilisé pour la gestion de liaison. Certaines trames de type U contiennent un champ d'information.

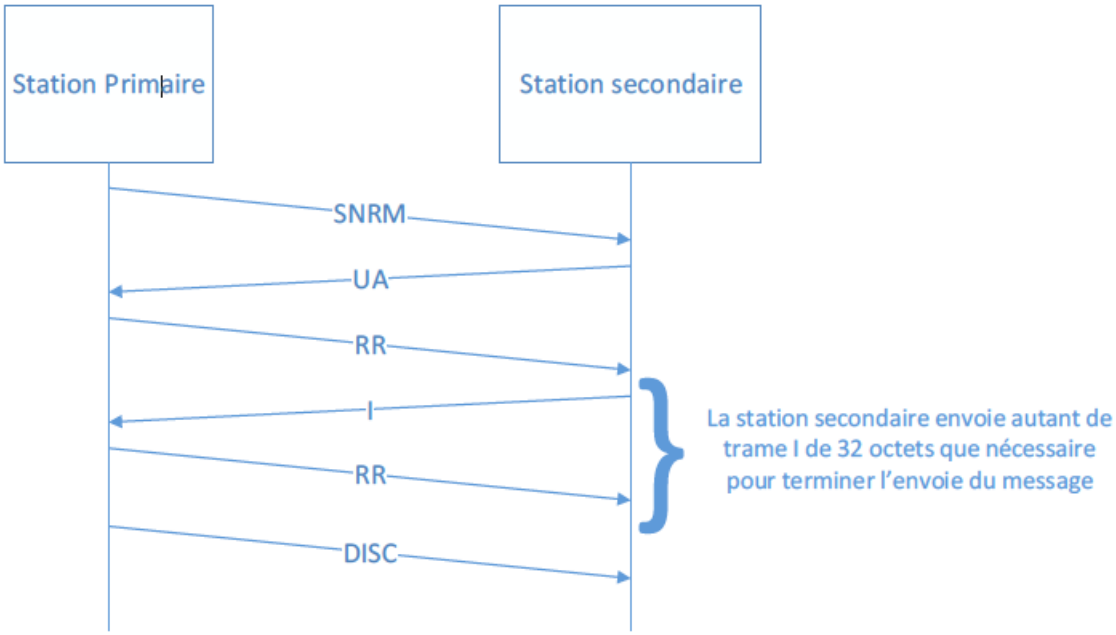
Supposition

Nous supposons que les trames sont envoyées sans erreurs. Dans le laboratoire la reprise des erreurs n'était donc pas considéré ni implémenté. Ceci a fait que le champ FCS n'est pas inclus dans la trame-I et les trames REJ et SREJ ne seront pas utilisées.

Algorithme

La station primaire fait une demande pour se connecter aux différentes stations secondaires (SNRM), soit 2 dans notre cas et attends de recevoir un acquittement des stations secondaires (UA) pour établir la connexion. La station primaire envoie ensuite un message disant qu'elle est prête à recevoir des données (RR). Ensuite, la station secondaire lui envoie une première trame-I de 32 octets provenant du message coupé en morceaux, puisqu'il est plus grand que 32 octets. La station primaire répond ensuite avec un acquittement (RR), disant qu'elle a bien reçu la trame et qu'elle est prête à recevoir la prochaine. Cette boucle s'effectue tant que le message de la station secondaire n'est pas complètement envoyé et qu'il reste un numéro de séquence disponible. Les stations sont capables de savoir la progression de l'envoi de données à l'aide des fenêtres coulissantes et des numéros d'acquittement lors de l'envoi d'un RR vers la station secondaire. Cette dernière va donc ajuster la fenêtre coulissante au nombre approprié de trames à envoyer. Lorsque l'envoie du message est terminé, la station primaire va envoyer une trame

de déconnexion (DISC) pour couper la connexion entre la station secondaire et elle-même.



Travail demandé

Afin de faire fonctionner le transfert de message en utilisant la fenêtre coulissante entre les stations secondaires et la station primaire, vous devez compléter le code de la classe SecondaryHDLCDataLink.

Dans cette classe, vous avez à compléter les méthodes suivantes :

- **dlDataRequest(String sdu) :** permet à la station secondaire d'envoyer un message (sdu) à la station primaire avec plusieurs trames-I du HDLC. Cette méthode vérifie en boucle s'il lui reste des données à transmettre et si la station a reçu une trame-RR, puis elle envoie une trame-I vers la station primaire si la vérification est positive.

- **checkNr(int nr, int rhs, int sz) :** cette méthode permet de calculer et retourner le numéro de trames qui ont été reçues et acquittées, permettant de savoir de combien la fenêtre coulissante doit être déplacée.

- **getRRFrame(boolean wait) :** cette méthode attend de recevoir une trame, pour ensuite vérifier si cette trame est bien de type RR. Si c'est de type RR, elle est retournée, sinon null est retourné.

dlDataRequest

Dans cette méthode on obtient d'abord une trame RR pour voir si la station est prête à recevoir. Ainsi, dès qu'on reçoit une trame indiquant cela, on envoi le SDU qu'on a reçu en paramètre.
Cette chaîne de caractère doit être divisée en plusieurs trames afin de l'envoyer à la station primaire, donc il faut commencer par la disposer dans un tableau de chaîne de caractère. Avec chaque chaîne de caractère du tableau il faut créer une trame I à envoyer.
Dans le cas de ce laboratoire il s'agit d'une trame asynchrone puisque l'on rajoute un bit pour la synchronisation de la trame. Donc, la trame qu'il faut créer doit être composée d'un drapeau, puis de l'adresse de la station, de l'indication qu'il s'agit d'une trame de données soit une trame I, il faut indiquer ensuite le numéro de séquence, puis placer la chaîne de caractère à envoyer, ainsi qu'un autre drapeau.

Il faut ensuite ajouter la trame au « frameBuffer » et incrémenter l'index ainsi que le numéro de séquence puisqu'on se prépare à envoyer la prochaine trame.

Par la suite, il faut utiliser la couche physique pour transmettre la trame I que nous avons créée. Après avoir envoyé la trame à la station on demande une trame de confirmation de la réception de la part de la station en question, c'est-à-dire une trame RR.

Lorsque cette trame est reçue, il faut convertir le champ nr, qui indique le prochain numéro de séquence que la station s'attend à recevoir, en nombre décimale. Puis, calculer le nombre de trames acquittée par la station en utilisant la fonction « checkNr » avec comme paramètre le numéro de confirmation que vient d'être calculer, une variable qui contient le numéro de séquence à droite du dernier numéro que l'on peut utiliser ainsi que la taille de la fenêtre.
De plus, grâce au calcul du nombre de trames acquittées on peut retirer ces trames du « frameBuffer » puisqu'on sait que la station les a belle et bien reçut et qu'on n'a pas besoin de les garder en mémoire pour les renvoyer.

Finalement, il faut bouger la fenêtre en calculant la nouvelle valeur du numéro de séquence à droite du dernier numéro que l'on peut utiliser dans la trame. On fait ceci, en additionnant le nombre de trame acquittée avec la dernière valeur de cette variable puis on fait le modulo avec la taille puisque c'est circulaire.

Ces étapes sont répétées plusieurs fois afin d'envoyer chaque chaîne de caractère du tableau qui a été créer précédemment. Enfin, la fonction retourne une valeur afin d'indiquer que les trames on belle et bien été envoyées.

checkNr

Cette méthode détermine le nombre de trames reconnues. Dans cette méthode, on reçoit trois paramètres qui sont: la taille de la fenêtre coulissante, le numéro du prochain numéro de séquence qu'on s'attend à recevoir et le numéro de séquence du dernier numéro de la fenêtre coulissante.
Il faut commencer par calculer la valeur du numéro de séquence de la partie gauche de la fenêtre. Il existe deux cas, soit la partie gauche est inférieure à la partie droite et dans ce cas la partie gauche est équivalente au côté droit moins la taille de la fenêtre.
Dans le cas contraire, la partie gauche est équivalente à la partie droite plus le nombre de numéro de séquence moins la taille de la fenêtre puisque la liste des numéros de séquence est circulaire.
Une fois c'est calculé, il faut vérifier que le numéro du prochain numéro de séquence est dans la fenêtre. S'il y est, alors le nombre de trames acquittées est égal au numéro du prochain numéro de séquence attendu moins la partie droite.

getRRFrame

La méthode « getRRFrame » retourne une trame RR. Il faut utiliser d'abord la méthode getFrame avec comme paramètre la valeur booléenne « wait » que l'on a reçu en paramètres, afin de récupérer la trame.
Ensuite, si cette trame n'est pas nulle, il faut vérifier qu'il s'agit d'une trame de supervision. S'il s'agit d'une trame S il faut sortir de la boucle, sinon tant que « wait » et vrai et qu'on n'a pas reçu la trame de supervision la boucle est ré-exécutée. Enfin, une fois sortie de la boucle, la méthode renvoi la trame de supervision.

Cas de test

Pour tester le fonctionnement de votre classe vous il faut utiliser l'impression disponible à la fin du guide du laboratoire 3 et la comparer à l'impression de l'exécution de votre code.