

UNIVERSITÉ DE MONTRÉAL

IMPLÉMENTATION D'UNE VERSION SIMPLIFIÉE DU PROTOCOLE HDLC

PAR
MATHIEU PERRON
YUAN SONG

BACCALAURÉAT EN INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

TRAVAIL PRÉSENTÉ À ABDELHAKIM HAFID
DANS LE CADRE DU COURS IFT 3325
TÉLÉINFORMATIQUE

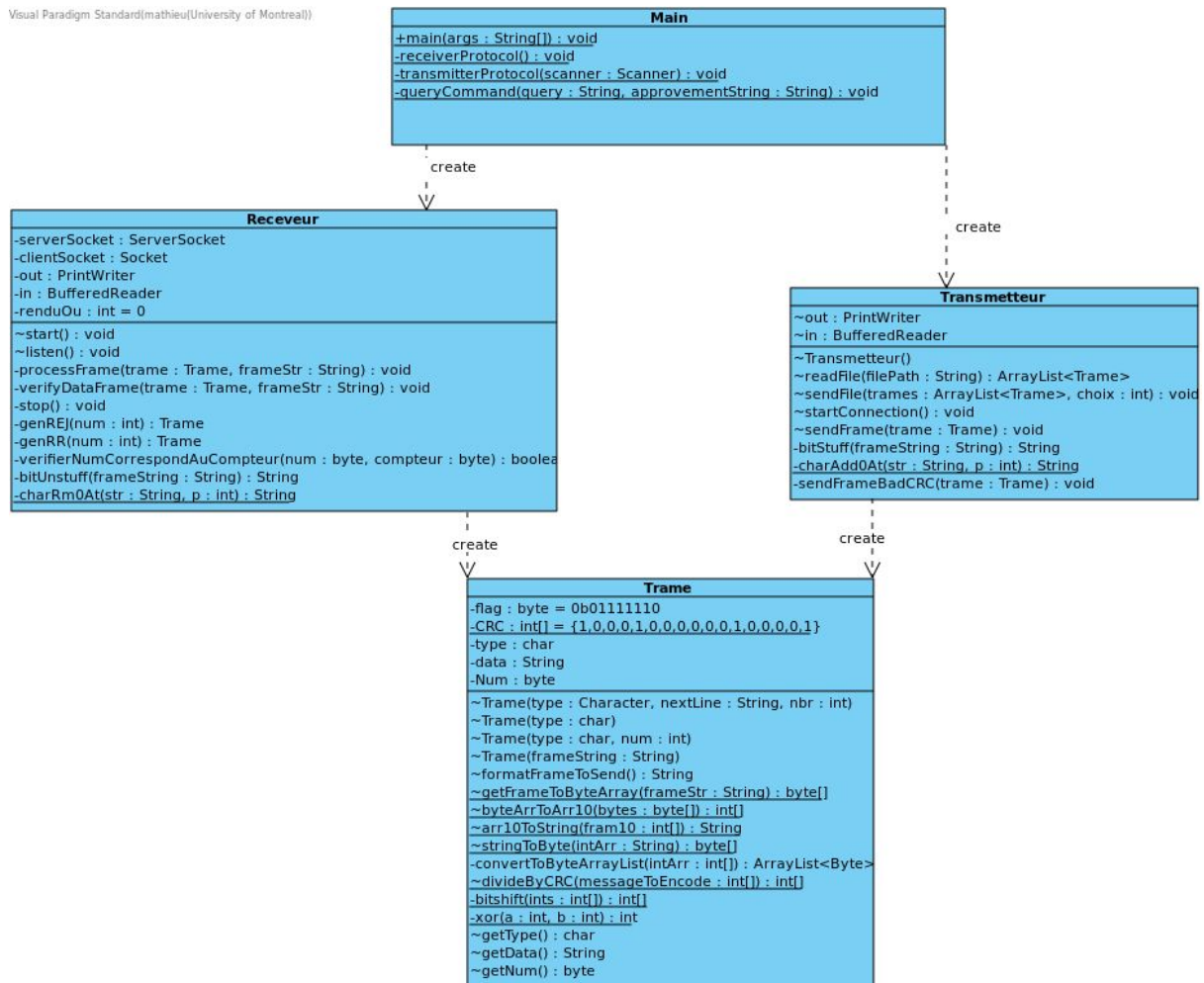
LE 30 NOVEMBRE 2019
SESSION AUTOMNE 2019

Table des matières

Diagramme de classes	3
Description des classes et méthodes	3
Trame	3
Transmetteur	5
Receveur	6
Main	7
3. Instructions pour la correctiowewew	8
4. Instructions pour la correction	8

1. Diagramme de classes

Visual Paradigm Standard(Mathieu(University of Montreal))



2. Description des classes et méthodes

L'architecture de notre implémentation est de type orientée-objet. C'est une architecture particulièrement adéquate pour le travail qu'on avait à accomplir puisque les entités utilisées dans le protocole sont facilement divisibles en des classes d'objets distincts. Voici un descriptif des classes et méthodes énumérées dans le diagramme de classe ci-dessus.

a) Trame

La classe Trame représente un objet trame. Elle est structurée selon la structure d'une trame tel que présenté dans l'énoncé, et ses propriétés sont énumérées dans le diagramme de classes. Concernant ses méthodes, on retrouve principalement des méthodes de formatage et de conversion d'un type à un autre. S'en suit une description du fonctionnement des principales méthodes de la classe. Il est à noter que les constructeurs des classes ne seront pas décrits, tout comme les méthodes accesseurs et modifieurs (get et set), puisque leur fonctionnement est trivial.

1. `String formatFrameToSend()`

Cette méthode permet de formater une trame à la fois, dans un format prêt pour l'envoi à l'interlocuteur (string). Elle prend en entrée deux `byte[]`, les données et le CRC calculé. Elle permet de formater ces `byte[]` en string de bits, qui sera envoyé à l'interlocuteur. La méthode crée un nouvel `ArrayList<Byte>`, lui ajoute les informations de la trame, et si la trame est de type I, calcule le CRC à partir du générateur en appelant la méthode `divideByCRC()` et la concatène. Elle appelle finalement `arr10ToString()` et retourne le résultat de l'appel, qui correspond à la trame convertie en string sous le format "11001010010101101...".

2. `static byte[] getFrameToByteArray(String frameStr)`

Cette méthode permet de convertir un string en entrée contenant les bits d'une trame en `byte[]`. Le `byte[]` est retourné.

3. `static int[] byteArrToArr10(byte[] bytes)`

Cette méthode permet de convertir un `byte[]` en entrée contenant les octets d'une trame en `int[]`, convertissant tous les octets en un flux de int de 0 et de 1. Le `int[]` est retourné.

4. `static String arr10ToString(int[] fram10)`

Cette méthode permet de convertir un `int[]` en entrée contenant les bits d'une trame en string contenant des caractères de 1 et 0. Le string est retourné.

5. `static byte[] stringToByte(String intArr)`

Cette méthode permet de convertir un `int[]` en entrée contenant les bits d'une trame en string contenant des caractères de 1 et 0. Le string est retourné.

6. `private static ArrayList<Byte> convertToByteArrayList(int[] intArr)`

Cette méthode permet de convertir un `ArrayList<Byte>` en entrée contenant les octets d'une trame en `int[]` contenant des int de 1 et 0. Le `int[]` est retourné.

7. `static int[] divideByCRC(int[] messageToEncode)`

Cette méthode effectue la division CRC par le générateur. Il prend en entrée un `int[]` correspondant aux données à encoder, rajoute les 0 à la fin, divise par le CRC. La méthode retourne un `int[]` correspondant au reste de la division.

8. `private static int[] bitshift(int[] ints)`

Cette méthode prend en entrée un `int[]` et effectue un bit shift vers la gauche (ce qui y correspond pour cette représentation des strings). Elle retourne le `int[]` résultant.

b) Transmetteur

Cette classe modélise l'objet transmetteur, qui correspond au client interagissant avec le serveur. Le transmetteur doit pouvoir :

- lire les données d'un fichier
- produire et envoyer des trames
- gérer les reçus du receveur
- ré-envoyer des trames en cas d'erreurs

Les attributs de la classe sont modélisés dans le diagramme de classes. S'en suit une description de ses méthodes.

1. `ArrayList<Trame> readFile(String fileName)`

Cette méthode sert à lire les lignes du fichier. Elle prend en entrée un string contenant le nom du fichier, et après avoir instancié un Scanner, lit chaque ligne à l'aide de celui-ci, la convertit en trame et rajoute au fur et à mesure la trame dans un ArrayList. La méthode retourne l'ArrayList contenant toutes les trames du fichier.

2. `void sendFile(ArrayList<Trame> trames, int choix)`

Cette méthode sert à envoyer des trames au receveur. Elle prend en entrée la liste de trames provenant du fichier et n'a pas de retour. Dépendamment de si on veut envoyer une trame en simulant une erreur ou non, elle envoie la trame erronée ou non jusqu'à temps que le compteur représentant la taille de la fenêtre du Go-Back-N soit de 0. Une fois le 0 atteint, on vérifie s'il y a eu une réponse du receveur ou non. S'il n'y a pas de réponse, le temporisateur fait attendre 3 secondes. S'il y a une réponse, on vérifie si la réponse est un RR ou un REJ, et on effectue le traitement correspondant.

3. `void startConnection()`

Cette méthode est appelée pour démarrer le transmetteur, et crée un nouveau socket servant de transmetteur. Le transmetteur doit être démarré seulement une fois que le receveur est démarré.

4. `void sendFrame(Trame trame)`

Cette méthode prend une trame en entrée, la traite avec le bit stuffing et envoie la trame stuffée en output au receveur.

5. `private String bitStuff(String frameString)`

Cette méthode effectue le bit stuffing dans le cas que le pattern du flag '01111110' se retrouvait dans les données. Si c'est le cas, la méthode, qui prend en entrée le string de données, modifie le pattern en y ajoutant un 0 après le 5e 1. Elle retourne le string modifié.

6. `private static String charAdd0At(String str, int p)`

Cette méthode sert à ajouter un caractère 0 à un string. Elle est appelée par la méthode traitant le bit stuffing. Elle prend en entrée le string et la position à laquelle elle effectue l'ajout.

7. `private void sendFrameBadCRC(Trame trame)`

Cette méthode a le même fonctionnement que la méthode `sendFrame()`, mais appelé avec un mauvais CRC. Elle sert à simuler le cas où il y aurait une erreur avec les bits transmis.

c) Receveur

Cette classe modélise l'objet receveur, qui correspond au serveur interagissant avec le client. Le receveur doit pouvoir :

- recevoir des trames
- vérifier la présence d'erreurs dans les trames
- produire et envoyer des reçus (RR)
- envoyer des REJ en cas d'erreurs

Les attributs de la classe sont modélisés dans le diagramme de classes. S'en suit une description de ses méthodes.

1. `void start()`

Cette méthode est appelée pour démarrer le receveur, et crée un nouveau socket servant de receveur. Le receveur doit être up avant le démarrage du transmetteur.

2. `void listen()`

Cette méthode est appelée pour permettre au receveur de commencer à recevoir des trames. Une fois la méthode appelée, le receveur commence à écouter pour des outputs d'un émetteur avec lequel il est connecté. Pour chaque trame reçue, il affiche un message à la console indiquant la réception, puis il appelle finalement la méthode `processFrame()` afin de traiter la trame.

3. `private void processFrame(Trame frames, String frameStr)`

Cette méthode sert à déterminer le type de traitement dépendamment du type de la trame reçue. Elle appelle la méthode correspondante au type pour le traitement/décodage. Les arguments en entrée sont la trame à traiter puis le string obtenu en input.

4. `public void verifyDataFrame(Trame trame, String frameStr)`

Cette méthode sert à valider le contenu d'une trame de type I reçue. Elle prend en entrée la trame et le string obtenu en input. Elle commence par convertir la trame en `byte[]`. Elle enlève ensuite les flags de l'array, puis effectue la division de CRC par le générateur (puisque le CRC n'est pas calculé sur les flags). La méthode vérifie ensuite qu'il n'y a pas d'erreur au niveau des bits ou de trames perdues. Dans le cas échéant, après 7 trames reçues correctes, elle renvoie un accusé de réception RR. S'il y a une erreur quelconque détectée, la méthode imprime l'erreur et envoie au transmetteur une trame REJ.

5. `private void stop()`

Cette méthode est appelée pour terminer la connexion établie entre le transmetteur et le receveur.

6. `private Frames genREJ(int num)`

Cette méthode génère une trame REJ à envoyer au transmetteur, en prenant en entrée le numéro de la trame à rejeter.

7. `private Frames genRR(int num)`

Cette méthode génère une trame RR à envoyer au transmetteur, en prenant en entrée le numéro de la trame reçu sans erreur.

8. `public boolean verifierNumCorrespondAuCompteur(byte num, byte compteur)`

Cette méthode sert à vérifier qu'il n'y a pas eu de trames perdues. Elle prend en entrée le NUM de la trame et le compteur, et retourne un booléen qui compare les deux octets. Puisque le compteur est incrémenté à chaque fois qu'une trame est reçue, si les deux octets n'ont pas la même valeur, une trame a donc été perdue. La méthode retourne true si les deux octets ont la même valeur, sinon false.

9. `public String bitUnstuff(String frameString)`

Cette méthode inverse le bit stuffing effectué lors de l'envoi de la trame. La méthode, qui prend en entrée le string de données, et traverse le string à la recherche du pattern dont le bit stuffing a été appliqué. S'il le trouve, il appelle la méthode `charRm0At()` afin d'enlever le 0 qui a été rajouté par le bit stuffing.

10. `public static String charRm0At(String str, int p)`

Cette méthode sert à enlever un caractère 0 à un string. Elle est appelée par la méthode inversant le bit stuffing. Elle prend en entrée le string et la position à laquelle elle effectue le retrait.

d) Main

La classe principale Main est la classe qui implémente le protocole. On y implémente une interface simple qui permet d'établir une connexion entre un transmetteur et un receveur, puis de choisir le cas de test. Dépendamment du choix entré, on y effectuera le transfert de trames avec ou sans erreurs. S'en suit une description de ses méthodes.

1. `public static void main(String[] args)`

La méthode Main implémente l'interface d'accueil initiale lors de la génération. À l'intérieur de cette méthode, on fait appel aux autres méthodes de démarrage du receveur et du transmetteur.

2. `private static void receiverProtocol()`

Cette méthode instancie un receveur et le rend prêt à recevoir des trames.

3. `private static void transmitterProtocol(Scanner scanner)`

Cette méthode instancie un transmetteur et le rend prêt à envoyer des trames. Elle fait envoyer les trames par le transmetteur, et selon le choix entré dans la console, elle appelle les méthodes de simulation d'erreurs ou non.

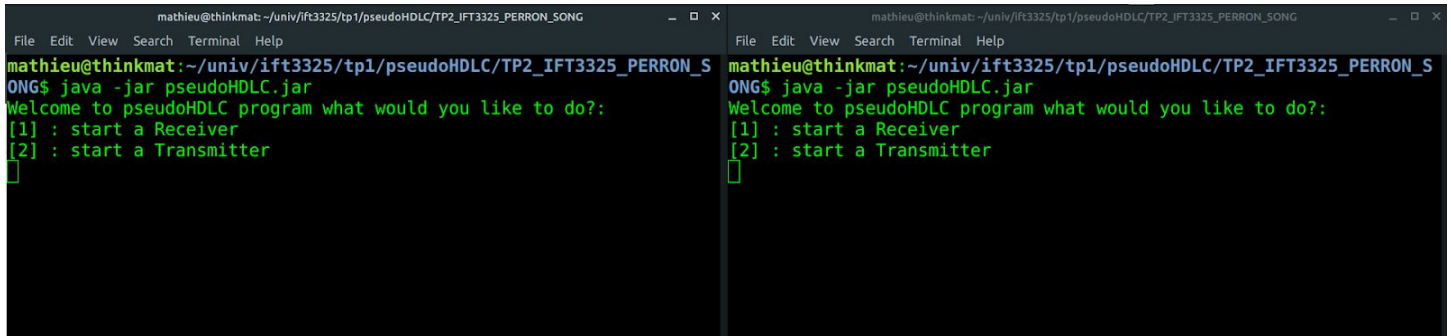
4. `private static void queryCommand(String query, String approvementString)`

Cette méthode instancie un scanner et dépendamment du choix entré en console, imprime un message passé en argument en console.

3. captures d'écran et fonctionnement du pseudoHDLC

Pour lancer le pseudoHDLC :

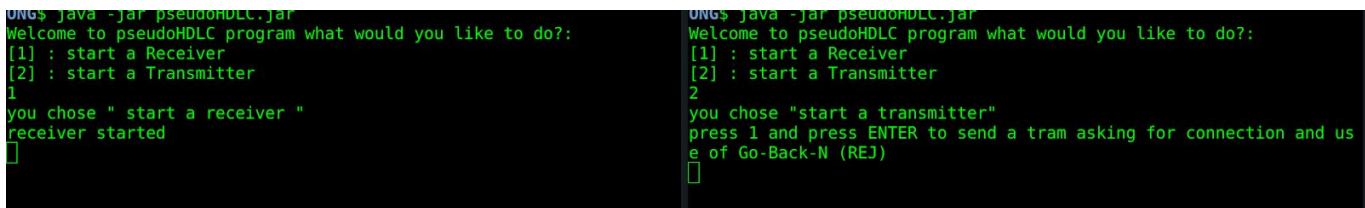
1. ouvrez deux fenêtres de terminal à l'emplacement du fichier jar pseudoHDLC.jar
2. dans chacune d'elles entrez la commande : `java -jar pseudoHDLC.jar`



```
mathieu@thinkmat:~/univ/ift3325/tp1/pseudoHDLC/TP2_IFT3325_PERRON_SONG$ java -jar pseudoHDLC.jar
Welcome to pseudoHDLC program what would you like to do?:
[1] : start a Receiver
[2] : start a Transmitter

```

3. Dans la première , entrez 1 , puis appuyez sur enter afin de démarrer le receveur.
4. dans la deuxième, entrez 2 puis appuyez su enter pour démarrer le transmetteur.



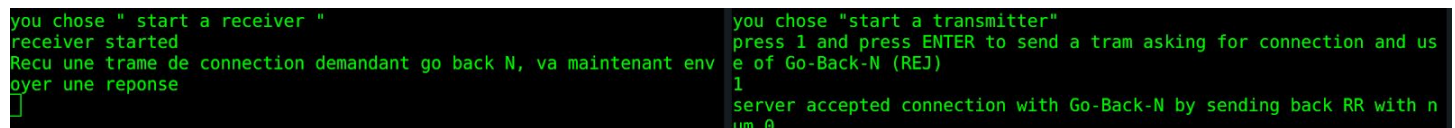
```

you chose " start a receiver "
receiver started

you chose "start a transmitter"
press 1 and press ENTER to send a tram asking for connection and use of Go-Back-N (REJ)

```

5. vous pouvez maintenant, dans le transmetteur, appuyer sur 1 pour envoyer la trame de demande de connection avec go back n.
6. le serveur la recevra et répondra avec la trame de confirmation



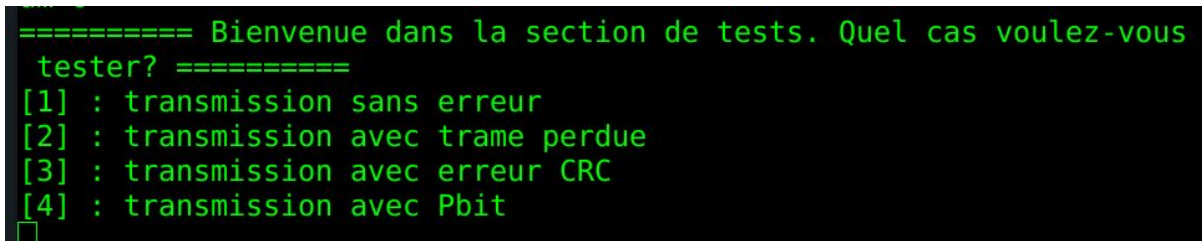
```

you chose " start a receiver "
receiver started
Recu une trame de connection demandant go back N, va maintenant envoyer une reponse

you chose "start a transmitter"
press 1 and press ENTER to send a tram asking for connection and use of Go-Back-N (REJ)
1
server accepted connection with Go-Back-N by sending back RR with num 0

```

7. un menu s'affiche maintenant à partir duquel vous pourrez sélectionner le test que vous voulez tester.



```

===== Bienvenue dans la section de tests. Quel cas voulez-vous tester? =====
[1] : transmission sans erreur
[2] : transmission avec trame perdue
[3] : transmission avec erreur CRC
[4] : transmission avec Pbit

```

8. si vous sélectionnez le premier , ce sera une transmission sans erreur:


```

you chose " start a receiver "
receiver started
Recu une trame de connection demandant go back N, va maintenant env
oyer une reponse
Recu du client: Trame avec num 0 contenant : Allo
Recu du client: Trame avec num 1 contenant : Bonjour
Recu du client: Trame avec num 2 contenant : test1$*
Recu du client: Trame avec num 3 contenant : test2!!
Recu du client: Trame avec num 4 contenant : test3
Recu du client: Trame avec num 5 contenant : Lorem ipsum dolor sit
amet, consectetur adipiscing elit.
Recu du client: Trame avec num 6 contenant : !!!!&&?*?$*
moment d'envoyer un RR
Envoi Trame RR 7 au client
Recu du client: Trame avec num 7 contenant : Baccalaureat
Recu du client: Trame avec num 0 contenant : ~Lorem~
Recu du client: Trame avec num 1 contenant : @test4
Recu du client: Trame avec num 2 contenant : test5
Recu du client: Trame avec num 3 contenant : test6
Recu du client: Trame avec num 4 contenant : test7
Recu du client: Trame avec num 5 contenant : test8
recu demande P de la part du client.
Envoie RR contenant 6
Recu du client: Trame avec num 6 contenant : test9
moment d'envoyer un RR
Envoi Trame RR 7 au client
Recu du client: Trame avec num 7 contenant : test10
Recu du client: Trame avec num 0 contenant : test11
Recu du client: Trame avec num 1 contenant : test12
Recu du client: Trame avec num 2 contenant : test13
Recu du client: Trame avec num 3 contenant : test14
recu demande P de la part du client.
Envoie RR contenant 4
nom du fichier?
test.txt
se prepare a l'envoi du fichier contenant 20 trames
Envoi de la trame 0 comportant le contenu : Allo
Envoi de la trame 1 comportant le contenu : Bonjour
Envoi de la trame 2 comportant le contenu : test1$*
Envoi de la trame 3 comportant le contenu : test2!!
Envoi de la trame 4 comportant le contenu : test3
Envoi de la trame 5 comportant le contenu : Lorem ipsum dolor sit a
met, consectetur adipiscing elit.
Envoi de la trame 6 comportant le contenu : !!!!&&?*?$*
serveur a repondu avec RR contenant num : 7
Envoi de la trame 7 comportant le contenu : Baccalaureat
Envoi de la trame 0 comportant le contenu : ~Lorem~
Envoi de la trame 1 comportant le contenu : @test4
Envoi de la trame 2 comportant le contenu : test5
Envoi de la trame 3 comportant le contenu : test6
Envoi de la trame 4 comportant le contenu : test7
Envoi de la trame 5 comportant le contenu : test8
Rien recu pour 3 secondes, envoie une trame de type P
serveur a repondu avec RR contenant num : 6
Envoi de la trame 6 comportant le contenu : test9
Envoi de la trame 7 comportant le contenu : test10
Envoi de la trame 0 comportant le contenu : test11
Envoi de la trame 1 comportant le contenu : test12
Envoi de la trame 2 comportant le contenu : test13
Envoi de la trame 3 comportant le contenu : test14
serveur a repondu avec RR contenant num : 7
Rien recu pour 3 secondes, envoie une trame de type P
serveur a repondu avec RR contenant num : 4
pour quitter [1]
pour continuer [2]

```

9. vous pouvez ensuite appuyer sur 2 puis entrée pour passer au prochain test

```

2
===== Bienvenue dans la section de tests. Quel cas voulez-vous
tester? =====
[1] : transmission sans erreur
[2] : transmission avec trame perdue
[3] : transmission avec erreur CRC
[4] : transmission avec Pbit

```

10. si vous réappuyez sur deux vous enclancherez le test qui générera une trame perdue. vous devrez entrer le nom du fichier test.txt comme avant qui doit se trouver dans le même path que votre fichier pseudoHDLc.jar

```

nom du fichier?
test.txt

```

11. le test 2 modifiera une trame et vous obtiendrez le comportement protocolaire souhaité

<pre> Recu du client: Trame avec num 0 contenant : Allo Recu du client: Trame avec num 1 contenant : Bonjour Recu du client: Trame avec num 2 contenant : test1\$* Recu du client: Trame avec num 3 contenant : test2!! Recu du client: Trame avec num 4 contenant : test3 Recu du client: Trame avec num 5 contenant : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Recu du client: Trame avec num 6 contenant : !!!!&?*?*\$* moment d'envoyer un RR Envoi Trame RR 7 au client Recu du client: Trame avec num 7 contenant : Baccalaureat Recu du client: Trame avec num 0 contenant : ~Lorem~ Recu du client: Trame avec num 1 contenant : @test4 Recu du client: Trame avec num 2 contenant : test5 Recu du client: Trame avec num 3 contenant : test6 Recu du client: Trame avec num 4 contenant : test7 Recu du client: Trame avec num 5 contenant : test8 recu demande P de la part du client. Envoie RR contenant 6 Recu du client: Trame avec num 6 contenant : test9 moment d'envoyer un RR Envoi Trame RR 7 au client Recu du client: Trame avec num 7 contenant : test10 Recu du client: Trame avec num 0 contenant : test11 Recu du client: Trame avec num 1 contenant : test12 Recu du client: Trame avec num 3 contenant : test14 Le compteur a détecté une trame manquante Envoie trame REJ 2 au client Recu du client: Trame avec num 2 contenant : test13 Recu du client: Trame avec num 3 contenant : test14 recu demande P de la part du client. Envoie RR contenant 4 </pre>	<pre> nom du fichier? test.txt se prepare a l'envoi du fichier contenant 20 trames Envoi de la trame 0 comportant le contenu : Allo Envoi de la trame 1 comportant le contenu : Bonjour Envoi de la trame 2 comportant le contenu : test1\$* Envoi de la trame 3 comportant le contenu : test2!! Envoi de la trame 4 comportant le contenu : test3 Envoi de la trame 5 comportant le contenu : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Envoi de la trame 6 comportant le contenu : !!!!&?*?*\$* serveur a repondu avec RR contenant num : 7 Envoi de la trame 7 comportant le contenu : Baccalaureat Envoi de la trame 0 comportant le contenu : ~Lorem~ Envoi de la trame 1 comportant le contenu : @test4 Envoi de la trame 2 comportant le contenu : test5 Envoi de la trame 3 comportant le contenu : test6 Envoi de la trame 4 comportant le contenu : test7 Envoi de la trame 5 comportant le contenu : test8 Rien reçu pour 3 secondes, envoie une trame de type P serveur a repondu avec RR contenant num : 6 Envoi de la trame 6 comportant le contenu : test9 Envoi de la trame 7 comportant le contenu : test10 Envoi de la trame 0 comportant le contenu : test11 Envoi de la trame 1 comportant le contenu : test12 (saute la trame 2 contenant test13 pour le test) Envoi de la trame 2 comportant le contenu : test13 Envoi de la trame 3 comportant le contenu : test14 serveur a repondu avec RR contenant num : 7 serveur a repondu avec REJ contenant num : 2 serveur doit renvoyer a partir de trame 2 Envoi de la trame 2 comportant le contenu : test13 Envoi de la trame 3 comportant le contenu : test14 Rien reçu pour 3 secondes, envoie une trame de type P serveur a repondu avec RR contenant num : 4 </pre>
--	--

12. en répétant les étapes précédentes vous pourrez lancer le test 3

<pre> Recu du client: Trame avec num 0 contenant : Allo Recu du client: Trame avec num 1 contenant : Bonjour Recu du client: Trame avec num 2 contenant : test1\$* Recu du client: Trame avec num 3 contenant : test2!! Recu du client: Trame avec num 4 contenant : test3 Recu du client: Trame avec num 5 contenant : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Recu du client: Trame avec num 6 contenant : !!!!&?*?*\$* moment d'envoyer un RR Envoi Trame RR 7 au client Recu du client: Trame avec num 7 contenant : Baccalaureat Recu du client: Trame avec num 0 contenant : ~Lorem~ Recu du client: Trame avec num 1 contenant : @test4 Recu du client: Trame avec num 2 contenant : test5 Recu du client: Trame avec num 3 contenant : test6 Recu du client: Trame avec num 4 contenant : test7 Recu du client: Trame avec num 5 contenant : test8 recu demande P de la part du client. Envoie RR contenant 6 Recu du client: Trame avec num 6 contenant : test9 moment d'envoyer un RR Envoi Trame RR 7 au client Recu du client: Trame avec num 7 contenant : test10 Recu du client: Trame avec num 0 contenant : test11 Recu du client: Trame avec num 1 contenant : test12 Recu du client: Trame avec num 2 contenant : test13 Une erreur a ete detectee dans le CRC Envoie Trame REJ 2 au client Recu du client: Trame avec num 3 contenant : test14 Recu du client: Trame avec num 2 contenant : test13 Recu du client: Trame avec num 3 contenant : test14 recu demande P de la part du client. Envoie RR contenant 4 </pre>	<pre> se prepare a l'envoi du fichier contenant 20 trames Envoi de la trame 0 comportant le contenu : Allo Envoi de la trame 1 comportant le contenu : Bonjour Envoi de la trame 2 comportant le contenu : test1\$* Envoi de la trame 3 comportant le contenu : test2!! Envoi de la trame 4 comportant le contenu : test3 Envoi de la trame 5 comportant le contenu : Lorem ipsum dolor sit amet, consectetur adipiscing elit. Envoi de la trame 6 comportant le contenu : !!!!&?*?*\$* serveur a repondu avec RR contenant num : 7 Envoi de la trame 7 comportant le contenu : Baccalaureat Envoi de la trame 0 comportant le contenu : ~Lorem~ Envoi de la trame 1 comportant le contenu : @test4 Envoi de la trame 2 comportant le contenu : test5 Envoi de la trame 3 comportant le contenu : test6 Envoi de la trame 4 comportant le contenu : test7 Envoi de la trame 5 comportant le contenu : test8 Rien reçu pour 3 secondes, envoie une trame de type P serveur a repondu avec RR contenant num : 6 Envoi de la trame 6 comportant le contenu : test9 Envoi de la trame 7 comportant le contenu : test10 Envoi de la trame 0 comportant le contenu : test11 Envoi de la trame 1 comportant le contenu : test12 (bousille le crc du frame 2 contenant test13 pour le test) Envoi de la trame 2 comportant le contenu : test13 Envoi de la trame 3 comportant le contenu : test14 serveur a repondu avec RR contenant num : 7 serveur a repondu avec REJ contenant num : 2 serveur doit renvoyer a partir de trame 2 Envoi de la trame 2 comportant le contenu : test13 Envoi de la trame 3 comportant le contenu : test14 Rien reçu pour 3 secondes, envoie une trame de type P serveur a repondu avec RR contenant num : 4 pour quitter [1] pour continuer [2] </pre>
--	---

13. puis le test 4....

4. Instructions pour la correction

Pour le correcteur, il est à noter que les fichiers source sont contenus dans le fichier .jar.